合刃科技创始人、华中科技大学教授　王星泽
美国麻省理工学院数学系助理教授　赵宇飞

联合力荐

附赠配套教学视频

少博士趣学

Python

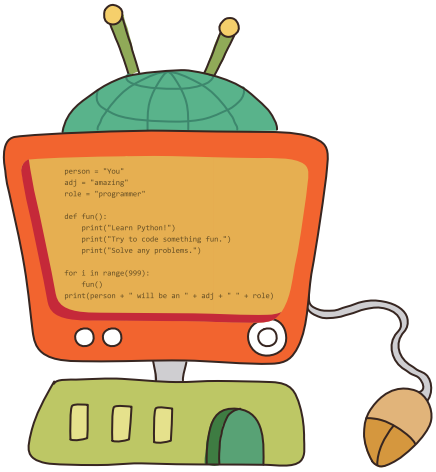
周安琪 / 编著

全彩印刷

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

少博士趣学
Python

周安琪 / 编著



電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

《少博士趣学Python》是一本编程与科技结合的启蒙书籍，全书从简单的小示例入手，介绍核心编程概念，并通过多个简单、有趣的编程案例，启发初学者探索身边的科技。例如，编写聊天机器人、扫地机器人、数学试卷机器人、绘图软件、密码程序等。书中不仅讲解了Python语法，还通过编程示例，给青少年读者以信息世界的启蒙。

图书在版编目（CIP）数据

少博士趣学Python / 周安琪编著. —北京：电子工业出版社，2019.2
ISBN 978-7-121-35461-8

Ⅰ. ①少… Ⅱ. ①周… Ⅲ. ①软件工具－程序设计－青少年读物 Ⅳ. ①TP311.561-49

中国版本图书馆CIP数据核字(2018)第254600号

责任编辑：李利健
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：720×1000　1/16　印张：15.25　字数：352 千字
版　　次：2019 年 2 月第 1 版
印　　次：2019 年 2 月第 1 次印刷
定　　价：79.90 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

推荐序 1

现在的孩子们是伴随着电子产品长大的，计算机作为学习或工作必备的电子产品，给他们带来了许多乐趣，其中有五花八门的游戏、各式各样的视频，甚至游戏化的学习软件。虽然这些活动都非常有趣，但始终只是作为消费者的一种乐趣。而如果孩子们学会了编程，自己创造游戏和动画，看着自己编写的程序可以运行起来，那么会有更高一层的乐趣和满足感。

对我而言，编程给我的童年带来了许多乐趣。在我 14 岁那年的暑假，出于好奇，我从父亲的书架上拿了一本编程书，开始自己探索编程。那是一个魔法般的新世界，学会使用代码后，我竟然可以更改游戏地图，重新创建游戏规则！这给我带来了极大的成就感。希望读者也可以跟着《少博士趣学 Python》探索编程的无穷乐趣。

计算机不仅有趣，还非常有用，它可以辅助人类解决许多问题。别看我的研究方向是数学和图论，但也和计算机科学有很大关联。学好了计算机科学和编程，就可以设计许多有用的算法，让我们的生活更加美好、便捷。我研究的图论数学可以帮大家更快地在互联网上搜索到想要的信息，帮物流公司更快地把商品送给大家等。能用自己学到的知识解决这些问题，让我非常兴奋。我希望读者在学习编程的过程中，也能思考如何利用所学的知识解决生活中的问题。

在学习编程的过程中，你肯定会遇到各种困难，不要担心，只要坚持探索和尝试，坚持练习，就一定能有很大的收获。最后，我建议大家和一两个小伙伴儿一起学，互相鼓励，互相交流，分享编程的乐趣。加油！

赵宇飞

美国麻省理工学院数学系助理教授

推荐序 2

热爱编程的小博士们，大家好！我是你们的博士同学王星泽。

你们知道每年 3 月 14 日是什么节日吗？是“圆周率节”。在历史上，关于圆周率的数值计算可以追溯到公元前 20 世纪，距离今天已经有 4000 多年的历史，包括中国古代数学家祖冲之在内的中外数学家们，为了得到圆周率小数点后的 N 位数值，倾尽了几乎一生光阴。

我难以忘记，在儿时的记忆中，第一次用计算机编程计算出圆周率近似值之后的喜悦心情！当时我用的语言是 C++，需要花费很大精力去调试程序；而现在你们很幸运，使用 Python 就可以轻而易举地得到古代数学家毕生钻研才得出的结果，更方便地体会到编程的乐趣。赶快去尝试一下吧！

如今，Python 成为人工智能时代的必学语言：它可以运行在 Windows、Mac、Linux、UNIX 等各种系统之上，我们可以轻松体验跨平台运行的乐趣；它具有简洁的特性和丰富的学习包，就像探索乐高玩具能拼出什么形状一样有趣、好玩；它可以被动态编译，也可以进行网络编程，还可以处理数据，是人工智能与深度学习领域中流行的语言。

古希腊著名科学家阿基米德说：给我一个支点，我将能撬动整个地球。而在今天，掌握了 Python 编程的技能，将为你开启一个全新的奇妙世界。这里有很多“思维实验”可以快速验证你的新想法，而伟大的创新大多都是通过这样的“思维实验”得来的。同学们，请投入 Python 的学习中，即刻验证你们大胆的想法吧！

王星泽

合刃科技创始人、华中科技大学教授

斯坦福大学博士、剑桥大学硕士、麻省理工学院学士

《福布斯》中国 30 岁以下 30 位科学家

《麻省理工科技评论》中国 35 岁以下 35 位科技创新青年

前言

少儿学编程该学什么

在多年的少儿编程教学过程中，我时常会思考这样一个问题：青少年学编程究竟应该学什么？有人说应该让孩子学会编写动画和游戏，让他们从消费者变成创造者；有人说未来科技人才短缺，需要从小培养顶尖的科技人才；有人说要锻炼计算思维能力和解决问题的能力。这些我都非常认同。

但我认为还有一点非常重要，那就是通过学习编程，了解我们这个以科技为核心的世界。我们需要领会计算机是如何快速进行重复性强、计算量大的工作，大范围地解决问题的。因为有了计算机，我们才能研究大量的基因信息，帮助医生治病救人；才能分析来自宇宙的大量数据，对未知世界进行探索；才能有自动驾驶及手机支付，让生活变得更加便捷。就像孩子们小时候会看百科全书了解身边的世界一样，他们也需要了解这个信息世界是如何运作的。同时，我们每天被科技“宠爱”着，依靠算法接收着我们想看的新闻、视频、产品和游戏，作为科技的消费者，我们的行为越来越多地被算法和数据所影响。只有了解它们，才能更清晰地面对这个以数据和科技为核心的世界。

本书特点

在《少博士趣学 Python》中，我希望教给读者的不仅是如何写 Python 代码，更是通过编写有趣的编程项目初识科技背后的故事。我们从简单的例子入手，逐渐增加编程项目的难度，通过不同的练习，思考身边的科技。本书并不是一本 Python 语法大全，Python 语言博大精深，而本书只接触到了冰山一角。

本书介绍了 Python 的许多入门知识，例如基本的语法、模块的使用，以及如何用 Tkinter 编写大家熟悉的图形化界面程序等。读完本书后，你就可以开始编写强大的程序了！

阅读对象

这是一本编程与科技结合的启蒙书籍，我并不想写一本针对资深极客或程序员的书，而是希望让更多的大朋友和小朋友通过这本书尝试编程。这本书适合：

- 想学编程的小朋友。
- 想教小朋友编程的老师。
- 想教小朋友编程的家长。
- 对科学技术好奇，想在轻松、有趣的环境下探索编程的大朋友。

当然，因为本书面向初学者，所以还有许多知识是书里没有讲到的，比方说制作游戏的 Pygame 模块、面向对象的程序设计方式等。我相信一名程序员应具备的能力之一是具有很强的学习能力，毕竟科技每几年都要更新迭代一次，一名好的程序员是有能力和动力去持续学习的。希望你也能够持续不断地学习，不断让自己进步！

如何使用本书

本书的每一章都经过了精心安排，在此建议初学者从头开始按顺序阅读，完成每个练习。另外，希望大家能够大胆尝试，改一改代码，看看修改过后的效果，在实践中学习。希望大家能够根据自己的创意和想象，编写出有趣的作品，帮助自己和身边的人解决问题。祝愿大家坚持学习，享受编程的乐趣！

本书为练习题提供了参考答案，读者可通过以下网址下载：

http://www.broadview.com. cn/35461

最后，我想特别感谢我的同事刘茗玉，她在我编写本书的过程中给了我莫大的帮助！

作　者

目录

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **下载资源**：本书提供配套资源文件，可在 下载资源 处下载。
- **提交勘误**：您对书中内容的修改意见可在 提交勘误 处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动**：在页面下方 读者评论 处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/35461

第 1 章
编程与我们的生活

本章重点是启发读者学习的兴趣，并对计算机在生活中的优点和缺点有所思考，希望读者带着兴趣和思考学习编程。

1.1 为什么学编程

1.1.1 通过编程了解以科技为核心的世界

在短短的十几年里，科技让我们以全新的方式生活、工作、娱乐。我们足不出户，便可以买衣服、买水果；我们相隔十万八千里，也可以通过视频面对面说话玩乐；我们可以随时随地找到各种视频、文章、游戏。

计算机能帮我们做重复性强、计算量大的工作。它能帮医生分析病人的大量基因信息，找到合适的疗法；它能帮汽车分析复杂的路况，找到合适的驾驶方案；它能分析我们在互联网上的动作，根据我们的喜好，推荐给我们喜欢读的文章、喜欢听的音乐和喜欢看的视频。

我希望你能通过学编程，了解我们这个以科技为核心的世界。并不是所有人都要成为专业的程序员，但科技已渗透到各行各业，了解和善用科技将至关重要。

1.1.2 编程很有趣

你想不想编写出自己的游戏和关卡?

你想不想编写电脑程序或者手机 APP?

你想不想制作酷炫的音乐、艺术作品、模型?

你想不想编写个人工智能机器人跟你下五子棋?

你想不想编写个程序自动给你喜欢的歌星投票?

你想不想编写个程序，自动每天早上提醒你今天的雾霾指数?

你想不想……

我相信你有许多妙趣横生的想法，现在我们就用程序将它们变成现实吧！

1.1.3 编程能锻炼你的思维能力

编程并不是一件容易的事，我们的程序也经常出错，需要我们耐心解决。编程可以挑战你的思维能力，我们将练习如何寻找规律；如何将庞大的问题分解成许多小问题，依次解决；面对同类问题，如何找寻系统性的解法（算法）；面对复杂的问题，如何隐藏不必要的细节，用简单的方式表达问题的核心。

我相信这些思维能力，不仅能用在编程上，也可以用在你生活的方方面面，让你更有效地思考与生活。

1.2 为什么学 Python

Python 简单易懂，接近英语表达且标点符号偏少，适合初学者学习。

Python 有许多强大的库，它的用途也很多，并且在数据处理、人工智能、机器学习方面应用广泛。

1.3 如何学好编程

恭喜你！能决定开始学编程，你已经迈出了一大步，要想学好编程，其实很简单，只要做到以下几点。

- 大胆学习新知识，尝试新想法。
- 遇到问题不要着急，慢慢地、一步一步地排查解决。
- 解决问题后，回顾自己学到了什么，还有哪些不足。
- 将你的所学讲出来。

不过，这个世界上最难的事，就是坚持不懈地做好简单的事。祝你坚持自主学习，持续进步！

1.4 计算机的长处和不足

在我们学编程之前，我们先打量一下我们的计算机。任何东西，无论是软件系统还是硬件物品，甚至是一幅画、一段乐曲，都有它的优缺点。习惯客观地分析优缺点，能帮助我们更有效地使用和改善它。

前面提到计算机非常有用。计算机有哪些优点，使它如此强大？

- **计算机的计算速度很快**，它计算的速度比人类快得多，可以快速处理大量数据。因为它快，才能帮医生分析大量的基因信息、帮企业家分析用户信息、帮自动驾驶系统分析路面信息。
- **计算机很精确**，只要给计算机正确的指令，它就能够不出错。有人会说，不是吧，我的电脑经常出错啊！这种情况大多数时候是因为人类给它的数据有问题，或给它的指令有误。如果你给计算机正确的指令，它就会正确地将任务完成。
- **计算机可以重复执行某些任务**，你可以让计算机重复执行某些任务千遍、万遍，它也不会抱怨，继续精确地执行着。而换作是人类，你让他重复做一件事，他就会感到无聊和疲惫。
- **计算机可以在危险的情况下工作**，比方说我们可以把机器人放到月球上帮助我们采集数据，进行计算。而我们很难让一个人长期住在月球上做同样的工作。

计算机有诸多优点，那它的不足又有哪些呢？我可以想到以下两点。

- **计算机会盲目遵守指令**，就算是错误的指令，它也会执行，给出错误的答案。
- **跟计算机沟通并不容易**，需要学习计算机的语言。

你还能想到计算机的哪些长处和不足？

我们学习编程，就是学习如何善用计算机的长处。如果哪个任务手动完成消耗时间长，重复性高，对精准度的要求高，那我们就用计算机来完成吧。

人开的车 VS 自动驾驶

- 你更愿意乘坐人开的车，还是软件系统开的自动驾驶车？
- 你更愿意乘坐连续开车 12 小时的人开的车，还是软件系统开的自动驾驶车？
- 你更愿意乘坐刚拿到驾照的人开的车，还是软件系统开的自动驾驶车？

结合计算机的长处和不足，讨论你对这三个问题的看法。

1.5　下载并安装 Python

Python 的官网是 www.python.org，我们可以直接从官网下载 Python。这里介绍在微软 Windows 和苹果 Mac OS 两种系统中的安装方式。

如果 Python 官网页面之后有所更新，那请大家用自己的思维能力和观察力，大胆地尝试，推测如何下载安装，解决问题。

1.5.1 Windows 系统

进入 https://www.python.org/页面，选择 Downloads，在弹出的菜单中选择 Windows 和 Python 3.x.y。我们的例子中是 Python 3.6.4。不同的数字代表不同的 Python 版本。需选择 3 开头的版本，后面两位数字并不重要。

下载 Python：在 Python 官网中选择【Downloads】→【Windows】→【Python 3.6.4】，如图 1-1 所示。

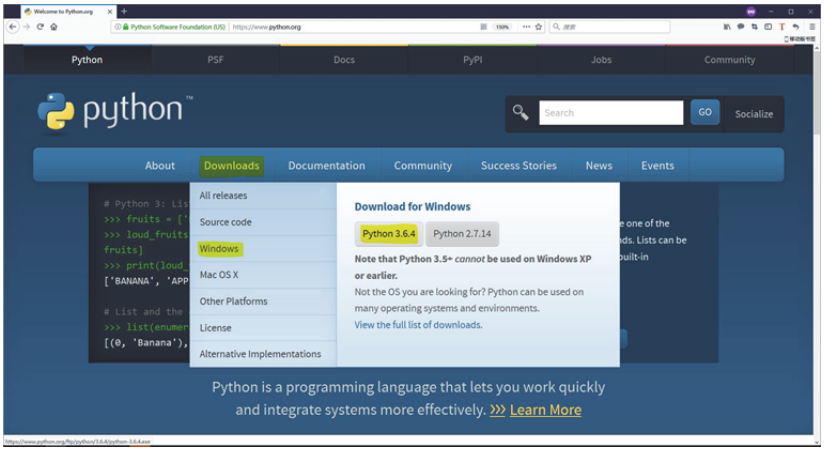


图 1-1　Python 官网下载

下载好后单击文件，选择 Install Now，根据提示安装。记住在这里要勾选 Add Python 3.6 to PATH，如图 1-2 所示。安装好后，打开 Windows 的开始菜单，找到 IDLE 程序。打开 IDLE 后，Python 的 Shell 窗口会弹出，如图 1-3 所示。准备就绪，可以翻到本书 1.6 节了解如何在 Shell 里编写代码。

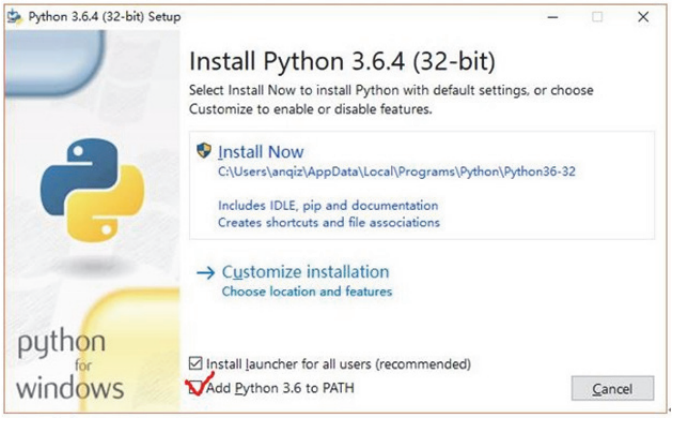


图 1-2　Windows 系统安装 Python

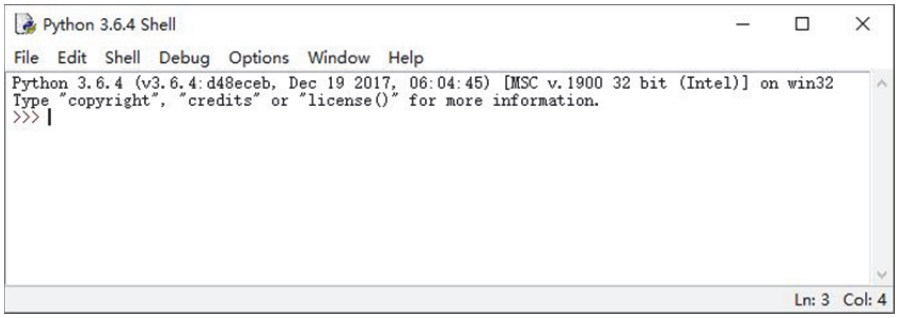


图 1-3　Python IDLE 页面

1.5.2　Mac OS 系统

如果你使用苹果电脑的 Mac OS 系统，下载时，请进入 https://www.python.org/页面，选择 Downloads，在弹出的菜单中选择 Mac OS X 和 Python 3.x.y。我们的例子是 Python 3.6.4。不同的数字代表不同的 Python 版本。需选择 3 开头的版本，后面两位数字并不重要。

下载 Python：在 Python 官网中选择【Downloads】→【Mac OS X】→【Python 3.6.4】，如图 1-4 所示。

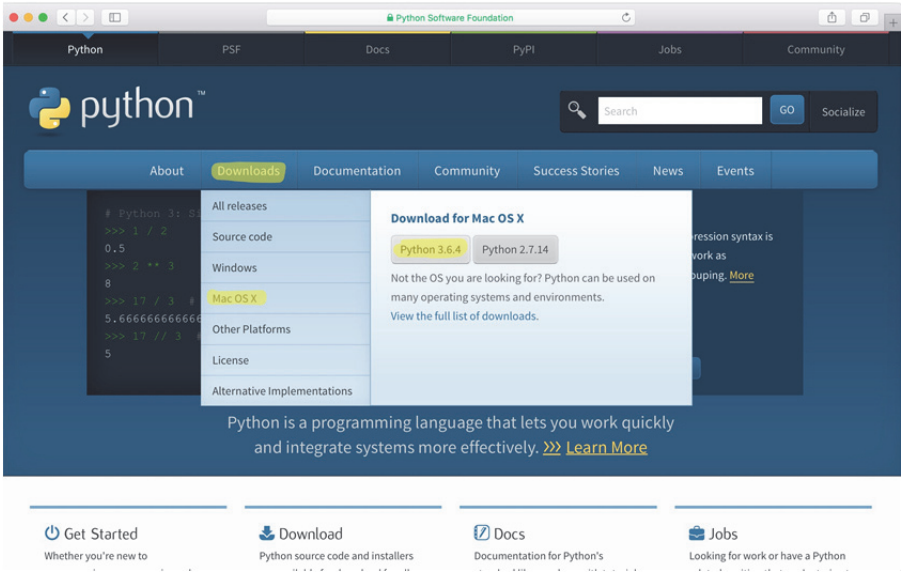


图 1-4　Mac 系统安装 Python

下载好后打开文件，根据提示安装。安装成功后，找到 IDLE 程序并打开 Shell 窗口，如图 1-5 所示。

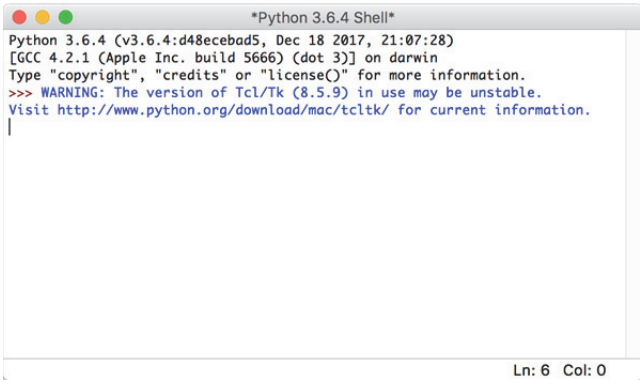


图 1-5　Mac 系统的 IDLE 页面

1.6　在 Shell 里编写代码

Python Shell 窗口自带交互功能，可以在其中直接输入 Python 代码运行。例如，在 Shell 窗口的 3 个箭头旁输入指令 2+3，按回车键，Python 自动计算出答案 5，如图 1-6 所示。

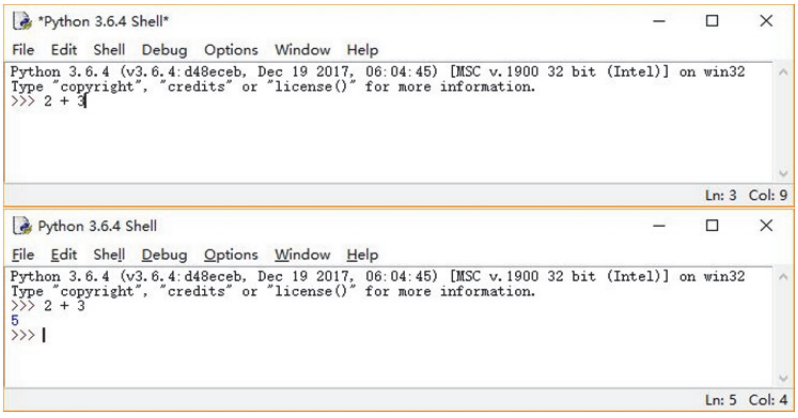


图 1-6　在 IDLE 输入指令

我们再尝试几个例子。

在 Shell 窗口里输入：

```
print("Hello, world")
```

“Hello, world”这句话会打印出来，如图 1-7 所示。“Hello, world”是许多人在学习编程时编写的第一行代码，广泛流传成为经典。print()是 Python 的打印函数，表示要打印出括号里的内容，在后面的章节里我们会经常使用 print()打印。

在编写程序时，要注意所有的 Python 标点符号比如引号“”、括号()都要用英文输入法，否则系统就会报错。另外，如果要找回之前的代码，只需要使用“Alt + P”组合键，就能自动补全之前打过的命令。

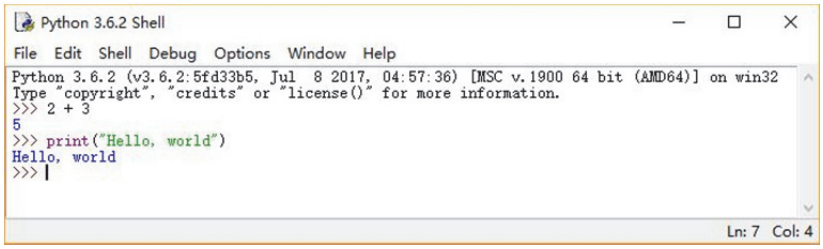


图 1-7　在 IDLE 里打印“Hello, world”

尝试在 Shell 窗口里输入：

```
for i in range(10):
    print("=^.^=")
```

注意，第二行要缩进，也就是说 print 前面需留有空格。事实上，只要第一行结尾处有冒号：在按下回车键后，Shell 窗口会自动空格。如果没有自动缩进，也可以手动敲两下或四下空格键缩进。编写完 print("=^.^=")后，按下回车键两次，结束编辑，Shell 窗口将打印 10 次小猫笑脸=^.^=，如图 1-8 所示。

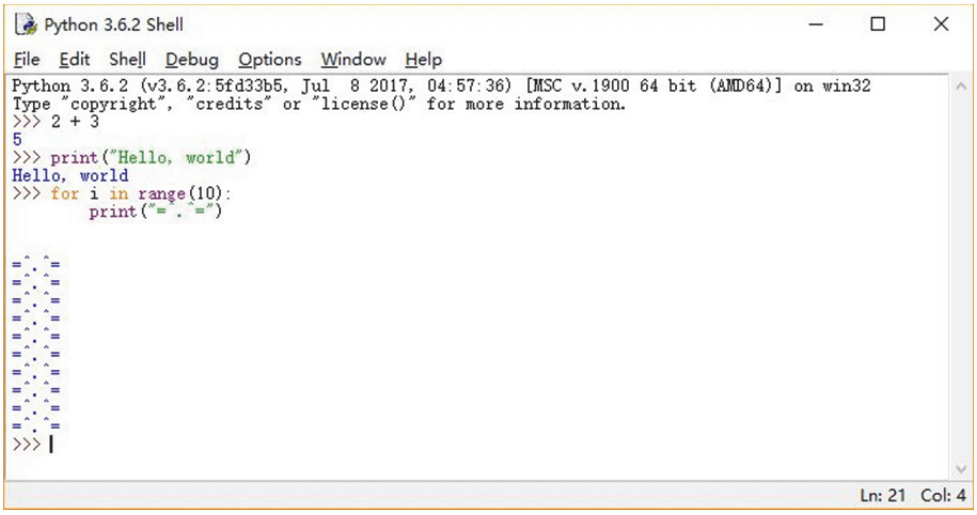


图 1-8　在 IDLE 里打印笑脸

在 Shell 中编写程序会很麻烦。例如，假设想打印 100 个笑脸，而不是 10 个，就需要重新写出这两行代码，十分不便。这时候就要请编辑器出场了。

1.7 在编辑器里编写代码

要想认真写代码，我们需要修改和存储功能，这就要依赖功能更加强大的编辑器了。

在 Shell 窗口里，点击左上角菜单中的 File，选择 New File，一个空白的编辑器窗口就打开了。

打开编辑器：Shell 窗口【File】→【New File】，在编辑器里输入下面的代码。

```
for i in range(100):
    print("=^.^=")
```

输入完后点击 File，选择“Save As...”，给 Python 代码文件起个名字，比方说“cat”，找到你想要保存的文件夹，比如桌面，选择保存。完成后一个名为 cat.py 的 Python 文件就保存在了你的桌面上，如图 1-9 所示。

我建议大家新建一个文件夹，专门存储你的 Python 学习文件，方便整理和回顾。

保存文件：编辑器窗口【File】→【Save As...】。

完成后，我们的文件标题变成了 cat.py，后面是这个文件的地址。

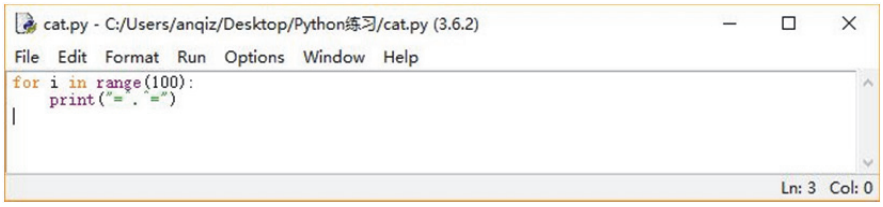


图 1-9 在 Python 的编辑器里编写程序

一切准备就绪，可以运行了，我们只要点击 Run，选择 Run Module，我们的 Shell 窗口就会运行代码，打印出 100 个小猫笑脸=^.^=。

运行 Python 文件：编辑器窗口【Run】→【Run Module】

大家在编写的时候，可将两个窗口并排摆放，方便编辑和测试，如图 1-10 所示。

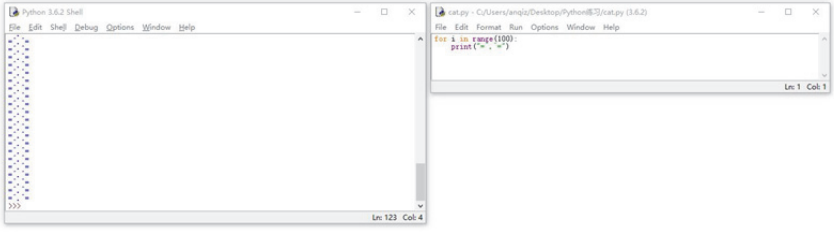


图 1-10 并排摆放 Python 编辑器和 Shell 窗口

之后若要打开 Python 文件，只要在 Shell 窗口或者编辑器窗口点击 File，选择 Open，找到想要打开的文件即可。

打开 Python 文件：Shell 窗口或编辑器窗口【File】→【Open】。

我们示范的是 Python 自带的 IDLE 软件。事实上，市场上有多个可用来编写 Python 的编辑器。各有优缺点。我们推荐的编辑器包括：

- 线上编辑器 Repl.it: https://repl.it/languages/python3
- Python 编辑器 PyCharm
- 编辑器 Sublime

大家感兴趣的话，可以探索这些编辑器，测试上面的 3 个例子，感受一下它们的优缺点。

1.8　五颜六色的代码

编辑器里的代码有多种不同的颜色，帮助我们读、写代码。每个编辑器的配色方案略有不同。所以如果你的代码颜色和我的不一样，不用担心。

但同一类型的代码颜色应该是一致的，比方说数字的颜色一致，引号包围的字符串颜色也应一致。如果有区别，可以根据提示检查代码是否出错。

```
for
in
10
print
range
"Hello, world"
"=^.^="
```

1.9　帮助我们的提示信息

就算是“大牛”程序员写出的代码也会出错。出错不要紧，我们的程序会给出提示。在图 1-11 的例子里，程序告诉我们问题出在第二行，priint 有问题呢！仔细看看，print 确实多写了一个 i。纠正过来就好了。

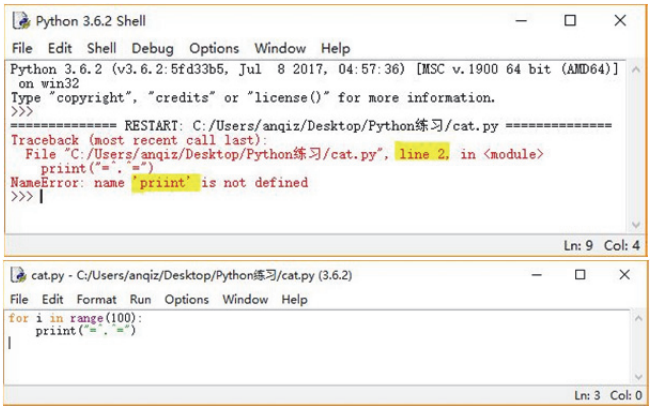


图 1-11　通过 Python 的报错提示找到代码中的问题

有了 Python 的提示，就算遇到 bug 程序报错，也不用担心，耐心纠错就好。毕竟，自己动手解决问题是快速学习新技能的不二法宝。

第 2 章

Python 编程初体验——发号施令

本章重点是让读者了解编程就是精确地发号施令，通过 Python turtle 感受简单的指令，熟悉 Python 代码。

2.1 什么是编程

编程是用***计算机的语言***，***精确***地给计算机发号施令，让它去完成某一件任务。

如果年龄太小，显示
不准来参加派对！

年龄太小是多小?

如果年龄<14，显示
不准来参加派对

请说计算机话！

```
age = input("你多少岁了？")
if age < 14:
    print("不准来参加派对！")
```

这些精确的指令，就是算法（algorithm）。有的算法可以帮我们给数字排队、有的算法可以做数学科学计算、有的算法可以美化图片、有的算法可以规划最短路线……

在什么情况下，使用什么算法，则是计算机世界的精髓。而把算法变成计算机的语言写出来，让计算机去执行，这就是编程。

2.2 给小海龟精确地发号施令

我们先跳过 Python 的语法，学一学如何精确地发号施令。

Python 有一个海龟库，又称海龟模块，用于制作有趣的图形。库里有只完全听指挥的小海龟，我们只要给它编程发号施令，它就会跟着指令画画。

2.2.1 指挥海龟画正方形

实例

我们先从简单的任务开始，指挥小海龟画个正方形，如图 2-1 所示。

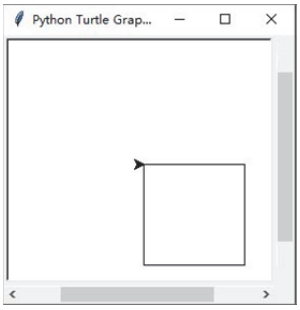


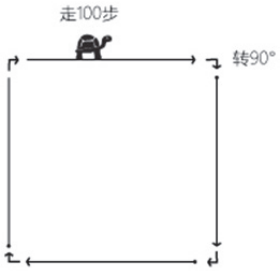
图 2-1 Python 海龟绘制正方形

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口:【File】→【New File】），给文件起个有意义的名字，保存到你的 Python 学习文件夹里（编辑器窗口:【File】→【Save As...】）。

若要调用海龟模块，首先需要编写 import turtle，然后用 turtle.Pen()让海龟现身，给它起个名字，我的海龟就叫 tony，你可以给你的海龟起其他任意名字。

```
import turtle
tony = turtle.Pen()
```

尝试运行这两行代码，小海龟现身了，但我们并没让它做其他的任务，所以它现完身程序就结束了。要想让小海龟画正方形，我们可以先思考它的算法是什么，然后编写出来。



画正方形的算法

向前走 100 步
向右转 90 度
向前走 100 步
向右转 90 度
向前走 100 步
向右转 90 度
向前走 100 步
向右转 90 度

确定好算法，我们可以开始编写程序。下面是一系列可以给小海龟下的指令。

指令	作用
import turtle	导入 turtle 模块
tony = turtle.Pen()	创建一个画布，给海龟起名为 tony
tony.forward(50)	让海龟 tony 往前移动 50 像素
tony.backward(50)	让海龟 tony 往后移动 50 像素
tony.right(90)	让海龟 tony 右转 90 度
tony.left(90)	让海龟 tony 左转 90 度
tony.circle(100)	让海龟 tony 画半径为 100 像素的圆形
turtle.bgcolor("black")	设背景颜色为黑色
tony.pencolor("red")	设海龟 tony 的线条颜色为红色
tony.width(2)	设海龟 tony 的线条粗细为 2 像素
tony.reset()	清除画布，让海龟 tony 回到开始的位置
tony.clear()	清除画布，让海龟 tony 留在原地

我们用 tony.forward(100)和 tony.right(90)为指令，组合出让小海龟画正方形的程序。

画正方形

1. 导入 turtle 模块
2. 给海龟起名为 tony
3. 向前走 100 步
4. 向右转 90 度
5. 向前走 100 步
6. 向右转 90 度
7. 向前走 100 步
8. 向右转 90 度

```
9. 向前走 100 步
10. 向右转 90 度
```

```
1. import turtle
2. tony = turtle.Pen()
3. tony.forward(100)
4. tony.right(90)
5. tony.forward(100)
6. tony.right(90)
7. tony.forward(100)
8. tony.right(90)
9. tony.forward(100)
10. tony.right(90)
```

点击【Run】→【Run Module】执行程序，就可以看到图 2-1 中小海龟画的正方形了。

2.2.2 指挥海龟画八边形

实例

根据画正方形的经验，尝试画如图 2 -2 的正八边形。

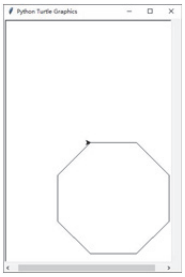


图 2-2 Python 海龟绘制八边形

程序员通常比较懒，但这种懒有好处，他们总在寻求以最快的方式完成任务，提高效率。为了不用重新写代码，程序员们可是想了许多种办法，后面我们会一一了解。能够在过去的解法里找出规律，套用到新的问题上，是一个重要的技能。让我们根据画正方形的解法，画八边形。首先，我们可以思考一下画八边形和正方形的主要区别，比方说旋转的角度有什么区别?

画八边形的解法和正方形基本类似，主要区别如下：

- 旋转的角度不同，需从 90 度改成 45 度

- 要画 8 条线，画线和旋转的次数从 4 次变成 8 次

基于正方形的代码，我们只要针对上面的区别修改一下就可以画出八边形。

下面就来尝试一下画正八边形吧！

画正八边形 1

```
1. 导入 turtle 模块
2. 给海龟起名为 tony
3. 向前走 100 步
4. 向右转 45 度
5. 向前走 100 步
6. 向右转 45 度
7. 向前走 100 步
8. 向右转 45 度
9. 向前走 100 步
10. 向右转 45 度
11. 向前走 100 步
12. 向右转 45 度
13. 向前走 100 步
14. 向右转 45 度
15. 向前走 100 步
16. 向右转 45 度
17. 向前走 100 步
18. 向右转 45 度
```

```
1.  import turtle
2.  tony = turtle.Pen()
3.  tony.forward(100)
4.  tony.right(45)
5.  tony.forward(100)
6.  tony.right(45)
7.  tony.forward(100)
8.  tony.right(45)
9.  tony.forward(100)
10. tony.right(45)
11. tony.forward(100)
12. tony.right(45)
13. tony.forward(100)
14. tony.right(45)
15. tony.forward(100)
```

```
16. tony.right(45)
17. tony.forward(100)
18. tony.right(45)
```

写好代码后点击【Run】→【Run Module】测试程序。

2.3 省力气的循环

我们八边形的代码中有许多重复的地方。tony.forward(100)和 tony.right(45)各重复了 8 次。聪明而懒惰的程序员们希望减少重复的代码。电脑的一大优点就是能帮我们完成重复性很高的工作，重复写同样的句子多没效率呀。因此，程序员们设计了循环语句，帮我们自动重复某些指令。

参考下面的代码，使用 for 循环，让 tony.forward(100)和 tony.right(45)这组动作自动重复 8 次。（我们将在第 7 章详细讲解循环语句。）

画正八边形 2

```
1. 导入 turtle 模块
2. 给海龟起名为 tony
3. 重复下面的代码 8 次：
    1）向前走 100 步
    2）向右转 45 度
```

```
1. import turtle
2. tony = turtle.Pen()
3. for i in range(8):
    → tony.forward(100)
    → tony.right(45)
```

我们的代码越来越复杂，许多代码前面有空格，这个空格叫“缩进”。可以通过按 4 下空格键进行缩进或者点击 Tab 键进行缩进。同样缩进的代码为一组，我们称这些组合在一起的代码为“代码块”。

在大段的代码里，我们将使用箭头→表示缩进。一个箭头→表示一级缩进，两个箭头→→表示二级缩进。

小贴士：循环语句

这个 for 循环将打印“早上好”8 次：

```
for i in range(8):
```

```
    print("早上好")

这个 for 循环将打印“下午好”3 次:
for j in range(3):
    print("下午好")

这个 for 循环将打印“晚上好”100 次:
for a in range(100):
    print("晚上好")
```

i、j、a 都是变量，用于存储数据。有关变量的详细内容请参考第 4 章，有关循环的详细内容请参考第 7 章。

写好代码后点击【Run】→【Run Module】测试程序，效果应和前面介绍的八边形一致。

2.4　旋转的正方形

学会了基础的指令，我们能够让小海龟做更有趣的图形了。

实例

我们先来尝试一下旋转的正方形，如图 2-3 所示。

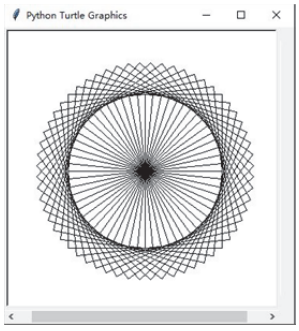


图 2-3　Python 海龟绘制旋转的正方形

根据循环语句小贴士和上一节的例子，尝试重复画 60 个正方形，每次画完后，再多旋转 6 度。

画 60 个正方形 1

1. 导入 turtle 模块
2. 给海龟起名为 tony
3. 重复下面的代码 60 次：
   1）向前走 100 步
   2）向右转 90 度
   3）向前走 100 步
   4）向右转 90 度
   5）向前走 100 步
   6）向右转 90 度
   7）向前走 100 步
   8）向右转 90 度
   9）向右转 6 度

```
1. import turtle
2. tony = turtle.Pen()
3. for i in range(60):
    → tony.forward(100)
    → tony.right(90)
    → tony.forward(100)
    → tony.right(90)
    → tony.forward(100)
    → tony.right(90)
    → tony.forward(100)
    → tony.right(90)
    → tony.right(6)
```

跟之前一样，我们的代码里 tony.forward(100)和 tony.right(90)各重复了 4 次。我们可以再用一个循环，进一步精简代码。

画 60 个正方形 2

1. 导入 turtle 模块
2. 给海龟起名为 tony
3. 重复下面的代码 60 次：
   1）重复下面的这组代码 4 次：
      a）向前走 100 步
      b）向右转 90 度
   2）再多向右转 6 度

```
1. import turtle
2. tony = turtle.Pen()
3. for i in range(60):
   → for j in range(4):
   → → tony.forward(100)
   → → tony.right(90)
   → tony.right(6)
```

注意，在第一个循环里我们用了循环变量 i，而在第二个循环里我们用了循环变量 j。在第 7 章里，我们会详细介绍循环的使用。在这里，我们先探索 for 循环的使用。

画完旋转的正方形后，我们可以尝试调整代码里的数字，看看是否能做出其他有趣的图形。

2.5　创造酷炫的图案

实例

我们每次循环时可以改变小海龟画的长度。for 循环里 i 的用处在于帮我们数数，从 0 数到 99，正好 100 次。与其让小海龟每次走 100 步，我们不妨尝试让小海龟走的步数正好跟数到的数字一样，画出酷炫的图案。

尝试下面的代码。

迷幻方形

1. 导入 turtle 模块
2. 给海龟起名为 tony
3. 重复下面的代码 100 次，从 0 开始数到 99：
   1）数到多少，就向前走多少步
   2）右转 90 度

```
1. import turtle
2. tony = turtle.Pen()
3. for i in range(100):
   → tony.forward(i)
   → tony.right(90)
```

写好代码后点击【Run】→【Run Module】测试程序，画出迷幻的图案，如图 2-4 所示。

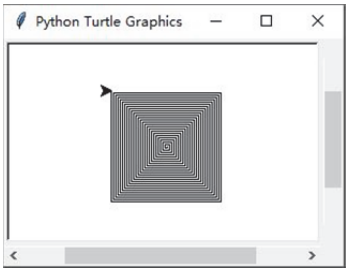


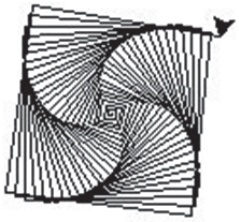
图 2-4　Python 海龟绘制迷幻的正方形

实例

每次旋转 90 度画出了基于正方形的迷幻图案，尝试将 90 改成 91、60、110，则会画出其他酷炫的图案。

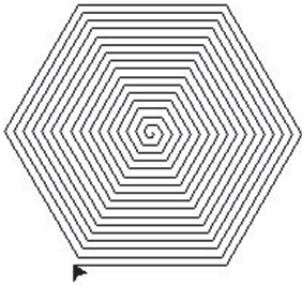
```
import turtle
tony = turtle.Pen()

for i in range(100):
    tony.forward(i)
    tony.right(91)
```



```
import turtle
tony = turtle.Pen()

for i in range(100):
    tony.forward(i)
    tony.right(60)
```



```
import turtle
tony = turtle.Pen()

for i in range(100):
    tony.forward(i)
    tony.right(110)
```

这些图形计算机最多几分钟就画出来了，可若要我用手画，先别说我会不会两分钟后就无聊到摔笔离去，就算我为了艺术有耐心慢慢创作，也难保我不会手轻轻一抖，不是角度量歪，

就是线条画歪。对于这种重复性高，精确度要求也高的任务，还是交给我们的计算机来完成好了。

2.6　给点颜色看看

若要让我们的图像更有趣，我们可以改变线条的颜色和粗细。

指令	作用
turtle.bgcolor(颜色字符串)	改变图画的背景颜色，bg 是英文 background 背景的缩写
tony.pencolor(颜色字符串)	改变小海龟 tony 的画笔颜色
tony.width(数字)	改变小海龟 tony 的画出线条的粗细

在这里，改变背景颜色指令的对象是 turtle，是整个屏幕，意思是我们要改变整个屏幕的背景颜色。而改变画笔颜色和线条粗细的对象是名叫“tony”的小海龟，意思是改变画出的这个小海龟 tony 的线条的颜色和粗细。

尝试编写下面的代码，改变图画颜色。

给点颜色看看

1. 导入 turtle 模块
2. 设定背景颜色为黑色
3. 给海龟起名为 tony
4. 设海龟画笔粗细为 3 像素
5. 重复下面的代码 100 次，从 0 数到 99：
   1）设画笔颜色为黄色
   2）往前走，步数为数到数字的 4 倍
   3）右转 96 度
   4）设画笔颜色为蓝色
   5）往前走，步数为数到数字的 4 倍
   6）右转 96 度

```
1. import turtle
2. turtle.bgcolor("black")
3. tony = turtle.Pen()
4. tony.width(3)
5. for i in range(100):
   → tony.pencolor("yellow")
   → tony.forward(i * 4)
   → tony.right(96)
   → tony.pencolor("blue")
```

```
    tony.forward(i * 4)
    tony.right(96)
```

写好代码后点击【Run】→【Run Module】测试程序，画出图 2-5 的彩色图案。

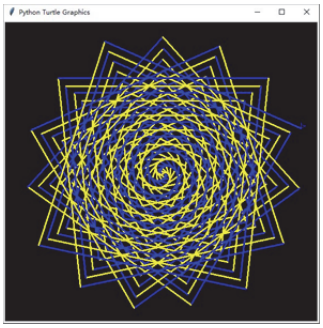
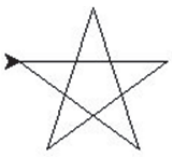


图 2-5　Python 海龟绘制彩色图案

2.7　总结及课后练习

在这一章节里，我们学习了如何给计算机精确地发号施令，并使用 Python 的 turtle 模块绘制出许多绚丽的图案。

1. 用 Python turtle 画出下面两种五角星。



a

b

2. 根据提示，编写代码画 100 个不同大小的圆圈，可以任意组合颜色及方向。

提示：tony.circle(100)可以让小海龟 tony 画出半径为 100 像素的圆形。

第 3 章

跟机器交流

本章重点是让读者通过 Python 的 input()和 print()了解计算机系统的输入、数据处理、输出，这 3 个步骤。

3.1 和计算机对话

通过上一章的学习，我们知道了编程是用***计算机的语言***，***精确***地给计算机发号施令，让它去完成某一件任务。我们可以通过编写代码，控制小海龟绘制出酷炫的图案。在本章中，我们将要学习使用 input()和 print()函数，使我们能和计算机对话。

让我们一起来尝试一个和计算机对话的练习。我们需要告诉计算机我们的名字，然后计算机会向我们问早上好。

实例

```
你叫什么名字? >>>诸葛亮
诸葛亮，早上好
```

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口:【File】→【New File】），保存好。

计算机打招呼 1

1. 询问“你叫什么名字?>>>”，并将得到的名字存储在 name 变量中
2. 打印 name 和“，早上好”。+表示连接变量 name 和字符串“，早上好”成为一句话

```
1. name = input("你叫什么名字? >>>")
2. print(name + ", 早上好")
```

点击【Run】→【Run Module】执行程序，就可以让计算机打招呼了。

实例

你叫什么名字？>>>犀利哥
犀利哥，早上好

在上面的例子中，我们使用了 input()和 print()函数，让我们一起来看一下它们的语法吧。

```
name = input("你叫什么名字？>>>")
```

input()函数等待用户在键盘上输入一些内容，按下回车键。在这个例子中，我们通过 input()函数得到用户的输入信息，将这个信息存储在名为 name 的变量中。

```
print(name + "，早上好")
```

print()函数将括号里的字符串打印在屏幕上。

小贴士：print()函数

1. print()默认输出是换行的，如果要不换行，可以使用 end 关键字参数，在末尾加上 end = ""。

```
print("a")
print("b")
print("c")

a
b
c

print("a", end="")
print("b", end="")
print("c")

abc
```

2. 在 print()函数中，使用加号“+”或逗号“,”都可以进行字符串的拼接。逗号连接会在两者之间多增加一个空格。

```
print("a" + "b" + "c")
print("a", "b", "c")
```

```
abc
a b c
```

3. 使用加号“+”拼接时符号两边需要用同类型数据。而使用逗号“,”拼接时，两边可以是不同类型的数据。

```
print("a" + "b" + "c")
print(1 + 2)

abc
3

print("a", "b", "c")
print("a", True, 1 + 2)

a b c
a True 3
```

4. 使用 sep 关键字参数，可以替换默认的分隔字符串。

```
print("a", "b", "c", sep="|")
a|b|c
```

5. 单、双引号用法一样，若字符串里需要保留引号使用时，可在需要保留的引号前加上“\”。三引号可以进行多行书写。

```
print('abc')
print("abc")
print('ab\'c')
print("ab\"c")
print('''a
b
c''')
abc
abc
ab'c
ab"c
a
b
c
```

学习了 input()和 print()函数，让我们来进行一下练习。修改上一个程序，使它出现下方的效果。

实例

```
你叫什么名字? >>>史蒂夫
现在是早上、中午还是晚上? >>>晚上
史蒂夫，晚上好
```

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口:【File】→【New File】），保存好。

复制、粘贴上一个程序的代码，然后进行修改。

计算机打招呼 2

1. 询问用户名字，将用户输入的内容存储在 yourName 变量中
2. 询问现在时段，将用户输入的内容存储在 yourTime 变量中
3. 连接变量 yourName、“,”、变量 yourTime，以及“好”，存进变量 sentence 里
4. 打印变量 sentence

```
1. yourName = input("你叫什么名字? >>>")
2. yourTime = input("现在是早上、中午还是晚上? >>>")
3. sentence = yourName + ", " + yourTime + "好"
4. print(sentence)
```

点击【Run】→【Run Module】执行程序，确认正确执行。

恭喜你基本掌握了输入、输出函数。让我们再进行一个进阶练习，程序会询问用户今年几岁并告诉用户他将在哪一年满 100 岁。运行程序效果如下。

实例

```
请问你叫什么名字? >>>莎士比亚
请问你今年多少岁? >>>9
莎士比亚将在 2108 年满 100 岁。
```

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口:【File】→【New File】），保存好。

过多少年是 100 岁

1. 用 input()询问名字，将用户输入的内容存储在 name 变量中
2. 用 input()询问岁数，将用户输入的内容存储在 age 变量中
3. 计算得出哪一年到 100 岁，存到 year 变量中，因为 age 是字符串，所以要用 int()转换成整数
4. 打印谁将在哪一年满 100 岁。变量 year 是整数，需使用 str()函数，转换成字符串

```
1. name = input("请问你叫什么名字? >>>")
2. age = input("请问你今年多少岁? >>>")
3. year = 2017 + 100 - int(age)
4. print(name + "将在" + str(year) + "年满 100 岁。")
```

点击【Run】→【Run Module】执行程序。

在上面的例子中，我们使用了 int()和 str()函数，让我们一起来看一下它们的语法吧。

```
year = 2017 + 100 - int(age)
```

int()函数将括号内的值转换为整数。在上面的例子中，用户的答案自动以字符串形式存储在 age 变量中。在计算时，我们需要用 int()函数将 age 变量里所存储的值转换成整数。

```
year = 2017 + 100 - age
print(name + "将在" + str(year) + "年满 100 岁。")
```

str()函数将括号内的值转换为字符串。在上面的语句中，变量 year 的数据类型是整数。通过 str()函数转换为字符串后，再通过“+”与其他的字符串进行连接。

小贴士：数据类型转换

1. int()函数将括号内的值转换为整数。

```
age = int("10")
print( age + 10 )

20
```

2. str()函数将括号内的值转换为字符串。

```
month = str(10)
day = str(1)
print(month + "月" + day + "日是国庆节。")

10 月 1 日是国庆节。
```

3. float()函数将括号内的值转换为浮点数，也就是小数。

```
mark = float("58.5")
print( mark + 1.5 )

60.0
```

更多关于数据类型的知识，请参考第 4 章。

3.2 输入和输出

输入可以理解为向计算机输入内容，给计算机传递信息。计算机根据我们输入的信息进行计算和处理。然后把内容输出。输入、数据处理、输出是计算机的三个重要步骤。

输入的方法有多种，可以用键盘打字输入、可以用麦克风输入声音，也可以用摄像头输入图片和视频，这些都是输入。输出也有不同的种类，可以在屏幕上打印出来、可以连接打印机打印，也可以用扬声器播放，等等。

在例子中，我们用 input()输入信息，将信息进行处理后用 print()输出。

3.3 跟人对话——注释

我们使用计算机语言编写代码，让计算机读懂后根据我们的指令完成任务，代码是人跟计算机沟通的语言。而有时候我们也会在代码里添加注释，这些注释不是为了跟计算机沟通，而是为了跟人沟通。通常，程序员用注释告诉自己或者同伴这一段代码是做什么用的，你可以想象成代码里的笔记。

#是注释的标志，#后的文本是注释。计算机在运行时，看到#会自动跳过注释，忽略注释里的内容。

```
name = input("你叫什么名字? >>>")#name 是变量，存储输入的名字
```

注释还可以是多行的。多行注释用三个单引号 ''' 或者三个双引号 """ 将注释括起来。

```
'''
```

```
多行注释
可以使用三个单引号
'''

"""
多行注释
可以使用三个双引号
"""
```

3.4　案例：笑话制造机

我们来编写一个叫“笑话制造机”的项目，让用户随意输入一些信息，组合成一段荒诞的笑话。运行程序效果如下。

实例

```
欢迎来到笑话制造机！
给我一个人名>>>哥斯拉
给我一个地方的名字>>>厕所
给我一个动作或者技能>>>必杀技
给我一个动物>>>流氓兔
给我一个东西>>>番茄
给我一个可以穿的东西>>>蜘蛛侠套装
    我和哥斯拉今天去厕所照相。我们照了两张相片。本来第一张我们想打扮成会必杀技的流氓兔，结果照片出来我们看上去很呆滞。在第二张照片里，我们看上去像是穿着蜘蛛侠套装的番茄，100%我想要的效果！
```

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

笑话制造机

1. 用 print()打印欢迎语句
2. 用 input()询问人名，存到 name 变量中
3. 用 input()询问地名，存到 place 变量中
4. 询问技能，存到 skill 变量中
5. 询问动物，存到 animal 变量中
6. 询问东西，存到 thing 变量中
7. 询问衣物，存到 clothes 变量中
8. 结合变量名编写笑话，使用加号“+”连接字符串，存进 joke 变量中

9. 打印 joke 变量

```
print("欢迎来到笑话制造机！")
name = input("给我一个人名>>>")
place = input("给我一个地方的名字>>>")
skill = input("给我一个动作或者技能>>>")
animal = input("给我一个动物>>>")
thing = input("给我一个东西>>>")
clothes = input("给我一个可以穿的东西>>>")
joke = "我和" + name + "今天去" + place + "照相。我们照了两张相片。本来第一张我们想打扮成会" + skill + "的" + animal + "，结果照片出来我们看上去很呆滞。在第二张照片里，我们看上去像是穿着" + clothes + "的" + thing + "，100%我想要的效果！"
print(joke)
```

点击【Run】→【Run Module】执行程序，运行自己设计的笑话制造机。

3.5 总结及课后练习

在这一章节里，我们了解了计算机程序中的输入、数据处理和输出，学习了 input()和 print()函数。

编写程序：猜生日

你玩过下面这样的游戏吗?

你不用告诉我你的生日日期，把你的生日日期乘以 2 加上 5 的结果告诉我。我就可以知道你的生日日期。

分析：假设你的生日日期是 8，8×2+5=21。然后我来通过推算，计算出你的生日日期。如何计算呢？我可以进行逆运算(21-5)÷2=8，得出你的生日日期。那么现在我们来编写一个程序。

编写程序：我可以知道你的生日。

1. 用你的生日月份乘以 4

2. 上一步结果加上 9

3. 上一步结果再乘以 25

4. 上一步结果再加上生日日期

例如，假设你的生日是 1 月 30 号。1*4 结果为 4，4+9 结果为 13，13*25 结果为 325，

325+30 结果为 355。只要你输入计算机是 355，计算机就可以推算出你的生日日期是 1 月 30 日。

运行程序效果如下。

```
我可以知道你的生日
1. 你生日月份乘以 4
2. 再加上 9
3. 再乘以 25
4. 再加上你生日日期
结果告诉我是>>>355
你的生日日期是 1 月 30 日。
```

提示：/为除，//为整除，10/4 将得出 2.5，而 10//4 只会保留整数，得出 2。

第 4 章
数据的世界

本章重点是让读者思考信息世界的数据，为什么要分类，并了解不同的数据类型。

4.1 变量

在计算机语言中，变量是指在程序运行中，用于存储数据的一个抽象概念。它就像一个盒子，我们把数据存储在里面。我们可以新建一个盒子存储数据，可以改变里面的数据，也可以提取里面的数据使用。变量名就是指这个盒子上的标签。变量值就是指这个盒子里的内容。

我们来看看下面这个例子：

- 变量 money 代表钱包，变量值 10 代表钱包中有 10 元
- 变量 card 代表银行卡，变量值 1000 代表银行卡中有 1000 元

我们用这两个变量记录下面的操作。

操作	钱包	银行卡
钱包 10 元，银行卡 1000 元	money = 10	card = 1000
从银行往钱包里取钱 500 元	money = money + 500	card = card – 500
买礼物，花费 488 元	money = money – 488	
卖出编写的 APP，得到 1000 元		card = card + 1000
打印出来，看看还剩多少钱	print(money) →22	print(card) →1500

我们通过变量名 money 或者变量名 card 可以找到相应的值。

注意，在这里 money = money + 500 可能对初学者来讲看起来十分别扭，因为这个算式在数学里是不成立的。但是此等号并非彼等号，并不表示左边等于右边。在 Python 里，等号是赋值运算符，意思是将右边的值存入左边的变量中。遇到等号赋值运算符时，我们应先计算出等号右边的值，再将它存入左边的变量中，先右后左。

4.1.1　为什么要用变量

了解了变量是什么后，我们来思考一下，为什么要用变量？有下面几个原因：

1. 我们想要在程序中存储数据。例如，得分、时间、血量等。
2. 方便程序编写和修改。例如，使用变量名代替程序中常出现的值，在修改程序时只需要修改一次。
3. 程序运行效率提高。例如，计算机运行程序中，对比计算速度，访问变量速度更快。

4.1.2　变量名

变量名就是变量的名字。我们一般使用有意义的单词命名。这有什么好处呢？

变量名有意义，有助于我们读程序，例如，变量 a 与变量 name 相比，通过“name”我们可以推测它可能用来存储一个代表名字的字符串。

在 Python 语言中，变量名遵循以下规则：

1. 由字母、数字、下画线等组成。

```
name_girl_2 = input("你叫什么名字？ >>>")#第二个女孩的名字的变量
```

2. 第一个字符不能是数字。

```
>>> 8name = 1
SyntaxError: invalid syntax
```

3. 区分大小写。

```
>>> name = "a"
>>> Name = "b"
>>> print(name + " " + Name) #name 和 Name 是两个变量
a b
```

4. 不能将关键字作为变量名。

```
>>> help("keywords")#查看关键字，下面这些关键字都不能作为变量名
Here is a list of the Python keywords.  Enter any keyword to get more help.

False               def                 if                  raise
None                del                 import              return
True                elif                in                  try
and                 else                is                  while
```

```
as          except      lambda      with
assert      finally     nonlocal    yield
break       for         not
class       from        or
continue    global      pass
```

小贴士：Python 变量、函数命名规范

1. 单词全小写，由下画线连接各个单词。

```
user_name = input("你叫什么名字? >>>")#用户名字的变量
def to_add(first_number,second_number)#做加法的函数
```

2. 小驼峰式（little camel-case）：第一个单词首字母小写，其余单词首字母大写。

```
userName = input("你叫什么名字? >>>")#用户名字的变量
def toAdd(firstNumber,secondNumber)#做加法的函数
```

4.1.3 变量有多可“变”

变量就是随时可变的量。我们使用赋值运算符来改变变量的值，例如：

```
name = "哈姆雷特" #字符串“哈姆雷特”赋值给变量 name
```

了解变量后，我们来尝试几个小练习。运行下面的程序，看看会打印出来什么呢?

变量练习 1

1. 右边的 9 赋值给变量 apples
2. 右边的 10 赋值给变量 bananas
3. 右边的 8 赋值给变量 apples，直接覆盖原有值
4. 打印变量 apples，结果是 8

```
1. apples = 9
2. bananas = 10
3. apples = 8
4. print(apples)
```

变量练习 2

1. 设变量 a 为 12
2. 设变量 b 为 15
3. 右边先提取变量 a 中的值为 12，再乘以 13，得到 156，再赋值给左边的变量 b
4. 打印变量 b，结果是 156

```
a = 12
b = 15
b = a * 13
print(b)
```

变量练习 3

1. 设变量 a 为 8
2. 右边先提取变量 a 中的值 8，再加 1，得到 9。再赋值给左边的变量 a
3. 打印变量 a，结果是 9

```
a = 8
a = a + 1
print(a)
```

等号的作用是赋值，我们计算出等号右边的数字后，赋值给等号左边的变量，也就是将新的值存入等号左边的变量中。

小贴士：赋值运算符 “=”

1. 遇到“=”时，先计算右边的值是多少，再把左边的变量设置为右边的值。

```
a = "ha" * 2
print(a)

haha
```

2. 可以使用连续赋值的方式，一次为多个变量进行赋值。

```
a = b = c = "ha" * 2
print(a,b,c)

haha haha haha
```

在这个例子里，我们用等号同时为 a、b、c 赋同样的值。

```
d,e,f = "cat","dog","fish"
print(d,e,f)

cat dog fish
```

在这个例子里，我们用逗号相隔，为 d、e、f 赋不同的值。

4.2 算法通过处理数据解决问题

在信息大爆炸时代，我们每天都会产生很多的数据。

我们在使用微信时会产生哪些数据呢?

- 你与朋友的用户名
- 头像图片
- 你发的消息：文字和语音
- 消息的时间
- 你的地点
- 钱包
- 视频

……

在微信的所有数据中，如果我们要找某些消息，可以输入关键词，使用微信搜索，搜索到相关消息记录。微信数据编码存储的方法会影响搜索的速度。我们当然希望搜索速度越快越好，那么我们就要采用好的算法来编码存储微信数据。

我们暂时不必知道哪种编码和搜索算法能让搜索更快，不过我们需要知道对数据进行分类是非常有必要的。

4.3 Python 数据类型及转换函数

4.3.1 常见数据类型

Python 3 中常见的数据类型包括 int（整数类型）、float（浮点类型）、bool（布尔类型）、string（字符串类型）、list（列表）、tuple（元组）、set（集合）、dictionary（字典）。

本节将详细介绍下面几个简单的数据类型。

- int（整数类型）：是指整数。比如，0、-1、15。
- float（浮点类型）：是指带小数点的数。比如，-1.3、0.0、3.14159265753。
- bool（布尔类型）：是指逻辑值，包含真（True）或假（False）两种值。
- string（字符串类型）：是指若干文本值。比如，"Hi"、"shanghai"、"苹果"。

小贴士：type()函数

type()函数可以用来查询变量的数据类型。

```
a, b, c = 1, 1.0, "hi"
print(type(a), type(b), type(c))
<class 'int'> <class 'float'> <class 'str'>
```

4.3.2　数据类型转换函数

在详细学习每一种数据类型的特点和使用方法之前，我们先了解下面的几个数据类型转换函数。例如，float(4)可以将整数 4 转换成浮点数 4.0。

函数	描述
int(x [,base])	将 x 转换为一个整数。如果 x 不是十进制数，则需要用第二个参数 base int("45")将返回 45 int("101101", 2)将返回 45，因为 101101 代表二进制数的 45
float(x)	将 x 转换到一个浮点数
bool(x)	将 x 转换成布尔值 如果 x 为 0 或 None 或 False 或空的值，则返回 False，否则它将返回 True
str(x)	将对象 x 转换为字符串

整数和浮点数都可以随意转换为字符串。整数和浮点数之间也可以任意转换。

```
>>> a = 12
>>> b = str(a)
>>> b
'12'
>>> a = 12.384
>>> b = str(a)
>>> b
'12.384'
```

```
>>> a = 12
>>> b = float(a)
>>> b
12.0
>>> a = "12.384"
>>> b = float(a)
>>> b
12.384
```

```
>>> a = 12.384
>>> b = int(a)
>>> b
12
>>> a = "12"
>>> b = int(a)
>>> b
12
```

小练习：a =“12”，怎样转换 a 才能让最终结果变成字符串类型的“12.0”？

转换成整数 int()、转换成浮点数 float()、换成字符串 str()。

```
a = "12"
a = str(float(a))
```

小贴士：数据类型转换

字符串不一定能转换为数字。例如，“string”不能被识别为数字。

```
>>> a =  "string"
```

```
>>> b = int(a)
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    b=int(a)
ValueError: invalid literal for int() with base 10: 'string'
```

小贴士：浮点数为什么不叫小数

在计算机程序中，为什么有小数点的数不叫小数，要叫浮点数?

这源于计算机编码存储小数的方式。在计算机中小数的表示方法有两种：定点数和浮点数。

定点数指小数点位置总是在数的某个特定的位置。例如，在银行系统中，小数点的位置总是在两位小数之前，这两位小数用来表示角和分。小数点后面只会有两位数，小数点的位置不会改变。计算机存储定点数，先确定小数点位置，然后再把小数点之前的数和小数点之后的数，编码为二进制存储。

浮点数是基于科学计数法的。计算机存储浮点数，先将数变成二进制数的科学计数法把数字转换成公式 $\pm M \times 2^E$。然后直接将 ±、M 与 E 转成二进制存储的三部分，这样就可以表示极大或极小的数。小数点的位置“漂浮”不定，故此得名“浮点”。

4.3.3 数据分类的好处

在 Python 语言中，将数据分成不同的数据类型有哪些好处呢?

- 能够更好地节省内存空间，不同的数据类型在内存中所占的字节数不同。
- 方便程序编译，计算机其实是使用二进制存储的机器。分成不同的数据类型，计算机会按照数据类型转换为计算机更容易操作的结构类型 01 二进制数。在 ASCII 编码中，字符大写 A 是 65（01000001），字符小写 a 是 97（01100001）。
- 方便程序运行，计算机会根据数据类型分配存储位置、确定作用域、做计算等。

4.4 数字

数字是我们非常熟悉的数据类型，包括整数和浮点数。

4.4.1 探索运算符

我们先来看看如何用熟悉的算术运算符对数字进行处理。

1. 加法运算符（+）、减法运算符（-）、乘法运算符（*）、除法运算符（/）

打开 Python Shell 尝试进行四则运算。

```
>>> 3 + 1
4
>>> 10 - 7
3
>>> 2 * 3
6
>>> 8 / 4
2.0
```

在这里我们有个问题要问问大家，为什么 8 / 4 的结果是 2.0 呢？你能想想吗？下一小节里有答案。

2. 取整除运算符（//）

拍脑袋可能想不出答案，最好的解决问题的方法就是大胆尝试，输入代码。我们何不在 Shell 里测试一下除号呢？事实上，Python 有两种除号/和//。在 Python Shell 中尝试下列代码，探索 / 和 // 的区别。

```
8 / 4
8 // 4
10 / 6
10 // 6
```

```
>>> 8 / 4
2.0
>>> 8 // 4
2
>>> 10 / 6
1.6666666666666667
>>> 10 // 6
1
```

根据测试结果，/和//都是做除法。使用/的除法结果是小数，而使用//的除法结果是整数，并且是向下取整，而非四舍五入。

3. 取模运算符（%）

在 Python 里%并不是百分比的意思。在 Python Shell 中尝试下方代码，探索%的作用。

```
8 % 4
9 % 4
10 % 4
11 % 4
8 % 5
9 % 5
```

```
10 % 5
11 % 5
```

```
>>> 8 % 4
0
>>> 9 % 4
1
>>> 10 % 4
2
>>> 11 % 4
3
```

```
>>> 8 % 5
3
>>> 9 % 5
4
>>> 10 % 5
0
>>> 11 % 5
1
```

根据测试结果，使用%运算符的结果会得到做除法后的余数。%是求余运算符，也叫取模运算符。我们会在程序里经常用到这个求余数的符号。

下表总结了 Python 的算术运算符。

运算符	描述	实例（a=10, b=21）
+	加，两个对象相加	a + b 输出结果 31
-	减，得到负数或是一个数减去另一个数	a – b 输出结果 -11
*	乘，两个数相乘或返回重复若干次的字符串	a * b 输出结果 210
/	除，将一个数除以另一个数	b / a 输出结果 2.1
//	取整除，返回商的整数部分	9 // 2 输出结果 4， 9.0 // 2.0 输出结果 4.0
%	取模，返回除法的余数	b % a 输出结果 1
**	幂，返回次方	a ** b 为 10 的 21 次方

除在数学中常见的运算符外，还有程序特有的一些符号，接下来我们看一看。

4. 加法赋值运算符（+= ）、减法赋值运算符（-= ）、乘法赋值运算符（*= ）、除法赋值运算符（/= ）

在 Python Shell 中尝试下方代码，探索 += 的作用。

```
a = 2
a += 1
print (a)
b = 10
b += 5
print (b)
```

```
>>> a = 2
>>> a += 1
>>> print(a)
3
```

```
>>> b = 10
>>> b += 5
>>> print(b)
15
```

根据测试结果，a += 1 相当于 a = a + 1，b += 5 相当于 b = b + 5。加法赋值运算符会先做加法运算再赋值给左边的变量。a += n 等效于 a = a + n。

在 Python Shell 中尝试下方代码，探索-=、*=、/= 的作用。

```
a = 2
a -= 1
print (a)
```

```
a = 5
a *= 2
print (a)
```

```
a = 20
a /= 2
print (a)
```

根据测试结果，a -= 1 相当于 a = a – 1，a *= 2 相当于 a = a * 2，a /= 2 相当于 a = a / 2。程序员总是喜欢“智慧的懒惰”，所以发明出这些赋值运算符帮助大家更快地编写程序。

下表总结了 Python 的赋值运算符。

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值给 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c – a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

了解了数字相关的运算符后，我们来通过编程练习，熟悉如何灵活使用这些符号，处理 Python 的数字。

4.4.2　案例 1：输出三位数中的每位数字

编写程序，用 input 让用户输入一个 3 位数的数字，分别打印出百位、十位和个位上的数字。

实例

```
请输入一个 3 位数>>>367
百位是 3
十位是 6
个位是 7
```

用户输入 367。百位数字使用 367 // 100 可得到 3。个位数字使用 367 % 10 可得到 7。另外，还要注意数据的类型！

先自己尝试一下这个编程练习吧，下面有具体的答案和解释。

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

输出三位数中的每位数字

```
1. 用 input()让用户输入三位数，用 int()转换为整数并赋值给变量 myNumber
2. 计算百位数字赋值给变量 bai
3. 计算十位数字赋值给变量 shi
4. 计算个位数字赋值给变量 ge
5. 打印百位数字
6. 打印十位数字
7. 打印个位数字
```

```
1. myNumber = int(input("请输入一个 3 位数>>>"))
2. bai = myNumber // 100
3. shi = myNumber % 100 // 10
4. ge = myNumber % 10
5. print("百位是" + str(bai))
6. print("十位是" + str(shi))
7. print("个位是" + str(ge))
```

点击【Run】→【Run Module】执行程序。

如果你想进一步挑战，那么编写程序用 input 让用户输入一个 4 位数的数字，分别打印出千位、百位、十位和个位上的数字吧。参考答案见本书附赠资料。

4.5 字符串

在这一小节里，我们将详细探索字符串及其相关的使用方法。

字符串可以由字母、数字、符号等组成。内容用英文的单引号或双引号括起来。

```
a = "ab 12.0 +="
```

4.5.1　字符串常见处理

1. 字符串的运算符

尝试下方代码，探索字符串的运算符加号“+”和运算符乘号“*”的作用。

```
a = "秦始皇大大"
b = "是我偶像"
c1 = a + b
c2 = a * 3
print(c1)
print(c2)

秦始皇大大是我偶像
秦始皇大大秦始皇大大秦始皇大大
```

尝试的结果表明，运算符加号“+”号可以连接字符串。运算符乘号“*”可以重复若干次字符串。

如果加号和乘号有效，那么减号“-”和除号“/”呢?

```
c1 = a - b
c2 = a / 3
```

结果报错，说明它们不能处理字符串。

2. 字符串里的索引值

要想找到字符串里的字符，需要用到索引值。我们先通过例子，观察索引值的使用方法。

尝试下面代码，看看结果是什么?

```
>>> a = "千里之行，始于足下"
>>> a[0]
```

尝试把 a[0]里的数字替换成下面的数字，探索它的作用。

```
>>> a[2]
>>> a[4]
>>> a[6]
>>> a[8]
```

```
>>> a[-1]
```

我们发现方括号里的数字将决定返回的字符。

```
>>> a = "千里之行，始于足下"
>>> a[0]
'千'
>>> a[2]
'之'
>>> a[4]
','
>>> a[6]
'于'
>>> a[8]
'下'
>>> a[-1]#负数是倒过来数
'下'
```

在字符串中，每个字符都有对应的索引值，代表它在字符串中的位置。如下表：

字符	千	里	之	行	，	始	于	足	下
索引值	0	1	2	3	4	5	6	7	8

在计算机的世界里，存储空间非常重要，所以我们从 0 开始计算，为了不浪费这个位置。我们通过索引值读取字符串中的某个字符。那如果我想读取一段字符呢？我们来尝试把 a[2]替换成下面代码，探索 a[::]的作用。

```
>>> a[1:3]
>>> a[1:8]
>>> a[1:7:2]
>>> a[1:7:3]
>>> a[1:8:3]
>>> a[:7]
>>> a[2:]
>>> a[2::3]
>>> a[8:1:-1]
>>> a[::-1]
```

字符	千	里	之	行	，	始	于	足	下
索引值	0	1	2	3	4	5	6	7	8

结果如下：

代码	打印	描述
a[1:3]	'里之'	提取字符串中索引值从 1 到 3（不包括 3）的字符
a[1:8]	'里之行，始于足'	提取字符串中索引值从 1 到 8（不包括 8）的字符
a[1:7:2]	'里行始'	每隔 2 字符，提取字符串中索引值从 1 到 7（不包括 7）的字符

续表

代码	打印	描述
a[1:7:3]	'里，'	每隔 3 字符，提取字符串中索引值从 1 到 7（不包括 7）的字符
a[1:8:3]	'里，足'	每隔 3 字符，提取字符串中索引值从 1 到 8（不包括 8）的字符
a[:7]	'千里之行，始于'	提取字符串中索引值从 0 到 7（不包括 7）的字符
a[2:]	'之行，始于足下'	提取字符串中索引值从 2 到最后的字符
a[2::3]	'之始下'	每隔 3 字符，提取字符串中索引值从 2 到最后的字符
a[8:1:-1]	'下足于始，行之'	倒着数每隔 1 字符，提取字符串中索引值从 8 到 1（不包括 1）的字符
a[::-1]	'下足于始，行之里千'	倒着数每隔 1 字符，提取字符串中的字符

我们可以在方括号里通过冒号提取字符串中的一段。根据上面的例子，我们总结出下面提取字符串的规律：

```
a[起始索引值 : 结束索引值后的一位 : 步长]
```

我们再来看看两个与字符串相关的函数。

len()函数得到字符串的长度

len(a)函数可以告诉我们字符串 a 中有几个字符。

```
>>> a = "吃葡萄不吐葡萄皮"
>>> len(a)
8
```

replace()函数替换字符串的子串

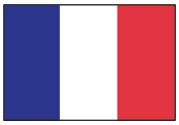
a.replace("葡萄","苹果")可以把 a 里面的“葡萄”换成“苹果”。

```
>>> a = "吃葡萄不吐葡萄皮"
>>> a.replace("葡萄","苹果")
'吃苹果不吐苹果皮'
```

在计算机的世界，经常需要对字符串进行处理，有时候需要压缩，有时候需要加密。下面我们给出几个相关的编程练习。

4.5.2　案例 2：国家名简写

学校打算举办二进制数比赛，有很多来自不同国家的选手参赛。每个国家的名字都用简写，简写规则是国家英文名的第 1、2、5 个字母。


FRANCE -> FRC


SPAIN -> SPN


SINGAPORE->SIA

让我们编写个程序，使用 input 让用户输入国家英文名，自动打印简写英文名。

实例

```
请输入国家英文名>>>FRANCE
FRC
```

提示：计算机从 0 开始计算，因此第 1 个字母的索引值是 0。

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

国家名代码 1

1. 用 input()让用户输入国家英文名，赋值给变量 countryName
2. 用加号连接变量 countryName 的索引值为 0、1、4 的字符，赋值给变量 codeName
3. 打印变量 codeName，显示简写国家名

```
1. countryName = input("请输入国家英文名>>>")
2. codeName = countryName[0] + countryName[1] + countryName[4]
3. print(codeName)
```

点击【Run】→【Run Module】执行程序。

有些国家的英文名没有第 5 个字母，所以我们打算换一种简写方式。新的简写规则是连接国家英文名的第 1、2 和最后一个字母。

编程练习：使用 input 让用户输入国家英文名，打印简写英文名。

实例

```
请输入国家英文名>>>FRANCE
FRE
```

提示：最后一个字母的索引值是什么呢？[-1]或者[字符串长度-1]。字符串长度可以使用 len()函数得到。

国家名代码 2

1. 用 input()让用户输入国家英文名，赋值给变量 countryName
2. 用加号连接变量 countryName 的索引值为 0、1、len(countryName)-1 的字符，赋值给变量 codeName
3. 打印变量 codeName，显示简写国家名

```
countryName = input("请输入国家英文名>>>")
codeName = countryName[0] + countryName[1] + countryName[len(countryName)-1]
print(codeName)
```

点击【Run】→【Run Module】执行程序。

在上面例子中，countryName[len(countryName)-1]与 countryName[-1]的结果是一样的。

什么要用 len(countryName)-1 作为索引值而不是 len(countryName)呢？假设 countryName 是“FRANCE”，len(countryName)会返回 6，而 countryName[6]却会报错。因为索引值从 0 开始，到 5 结束，所以我们需要用 len(countryName) -1 找到最后一个字符的索引值，也就是 5，对应 countryName[5]返回“E”。

4.5.3　案例 3：城市名加密

在打仗的时候，经常需要用密码来交流。我们和伙伴们约好了，每次进攻的城市要按照下面的方式加密。对长沙（changsha）加密后变成了 ehsgnehc。

编程例子：使用 input 让用户输入城市拼音，打印城市密码。

实例

```
请输入城市拼音>>>changsha
ehsgnehc
```

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

城市拼音加密

1. 用 input()让用户输入城市拼音，赋值给变量 city

2. 反转变量 city 中的字符串，赋值给变量 code
3. 将变量 code 字符串中的"a"替换为 "e"，赋值给变量 code
4. 打印变量 code，显示城市密码

```
1. city = input("请输入城市拼音>>>")
2. code = city[::-1]
3. code = code.replace("a", "e")
4. print(code)
```

点击【Run】→【Run Module】执行程序，看看能否得到预期的结果。

4.6 布尔值

介绍完字符串后，我们来看看布尔值。

看到 True 和 False 你能想到什么？英语判断题的对或错?

在 Python 中也有 True 和 False，它们往往是某判断句的结果。True 表示真，False 表示假。

```
>>> a = 2
>>> b = 2
>>> a == b #==是等于，a 等于 b 吗? True
True
>>> a = 3
>>> b = 2
>>> a == b
False
```

4.6.1 布尔值及底层的意义

True 和 False 是布尔值。19 世纪英国数学家 George Boole 发明使用 True 和 False 构成逻辑系统。True 和 False 可以直接赋值给变量，需要注意首字母大写。

```
>>> a = True
>>> b = False
```

在生产线上，机械手臂程序会用到布尔值：如果生产的零件是合格的（goodQuality 为 True），那么抬高机械手臂，让零件通过。如果生产的零件是不合格的（goodQuality 为 False），那么放下机械手臂，处理不合格的零件。

登录 QQ 时，验证密码程序也会用到布尔值：如果输入的 QQ 密码正确（correctPassword 为 True），那么让用户成功登录 QQ。而如果输入的 QQ 密码错误（correctPassword 为 False），那么提示密码错误，请用户重新输入。

在上述的例子里，都只有两种状态，合格或不合格，密码正确或不正确。这种只有两种状态的数据适合用布尔值表示。

4.6.2　比较数据

我们通过比较运算符比较不同的数据，并输出布尔值为结果。

在 Python Shell 中尝试编写下方比较数据的代码，观察结果。

```
2 > 3
5 * 5 >= 22
10 / 5 < 3
13 <= 18
4 == 5
8 != 9
```

```
>>> 2 > 3
False
>>> 5 * 5 >= 22
True
>>> 10 / 2 < 3
False
>>> 13 <= 18
True
>>> 4 == 5
False
>>> 8 != 9
True
```

根据测试结果，Python 使用>、>=、<、<=、==、!=这些符号来比较大小，并得到一个布尔值作为结果。

Python 中的比较运算符，总结如下：

运算符	描述	实例(a = 3,b = 5)
==	返回左边和右边是否相等	(a == b) 返回 False
!=	返回左边和右边是否不相等	(a != b) 返回 True
>	返回左边是否大于右边	(a > b) 返回 False
<	返回左边是否小于右边	(a < b) 返回 True
>=	返回左边是否大于或者等于右边	(a >= b) 返回 False
<=	返回左边是否小于或者等于右边	(a <= b) 返回 True

数字可以进行比较，那其他数据类型呢?

在 Python Shell 中使用 a == b 和 a != b 比较两组 a 和 b，观察结果。

```
a = 'I am an apple'
b = 'I am an orange'
a = [1,2,3,4,5]
b = [1,2,3,4,5]
```

```
>>> a = 'I am an apple'
>>> b = 'I am an orange'
>>> a == b
False
>>> a != b
True
```

```
>>> a = [1,2,3,4,5]
>>> b = [1,2,3,4,5]
>>> a == b
True
>>> a != b
False
```

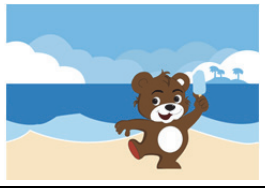
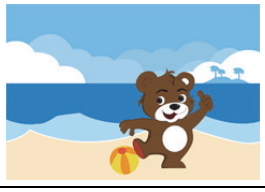
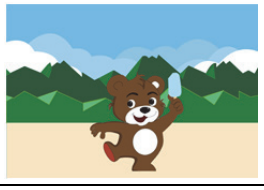
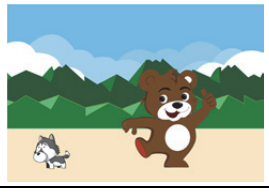
根据测试结果，字符串和列表可以使用==、!=运算符进行比较。至于其他比较运算符（>、>=、<、<=）是否可以用来比较字符串与列表，你可以在 Python Shell 中试一试。

4.6.3 布尔值与逻辑运算符的故事——小熊选照片

布尔值和逻辑运算符的概念其实在成长过程中已经潜移默化地植入我们的脑海里了。不信的话，我们做一下下面这个简单的练习题。

小熊霸霸去郊游，它拍了许多照片，想要选几张放进朋友圈，我们来帮它一起挑选吧。

小熊霸霸觉得海滩与帽子都是它喜欢的休闲元素，因此想选同时有海滩背景与戴着帽子的照片。以下哪些照片符合要求呢?

		
1	2	3
		
4	5	6

只有第 6 张照片同时满足有海滩背景和戴帽子这两个要求。

小熊霸霸觉得一张照片太少了，想了想说，其实这两个休闲元素只要有一个就够了，因此想选有海滩背景或戴着帽子的照片。那么又有哪些照片符合要求呢?

答案是第 1 张、第 2 张、第 3 张、第 6 张照片满足要求。

小熊霸霸想了想，其实它并不喜欢照片里的那只小狗，希望镜头中只有它自己，因此想选并非有小狗的照片。那么又有哪些照片符合要求呢?

答案是第 2 张、第 3 张、第 4 张、第 6 张照片满足要求。

4.6.4　逻辑运算符

在小熊选照片的练习中，出现了“与”“或”“非”。如果我们掌握了它们的意思，那么我们就能很快地找对照片了。

事实上这三个字在 Python 中对应着三种逻辑运算符，分别是：“与”（and）、“或”（or）、“非”（not）。我们来看看它们的意思。

1. 逻辑运算符——与（and）

```
>>> 3 > 2 and 1 > 0 #True and True 会返回 True
True
>>> 3 > 2 and 1 < 0 #True and False 会返回 False
False
>>> 3 < 2 and 1 > 0 #False and True 会返回 False
False
>>> 3 < 2 and 1 < 0 #False and False 会返回 False
False
```

逻辑运算符“与”（and）的左右两边都是 True，才会返回 True，只要有任意一边是 False，就会返回 False。

2. 逻辑运算符——或（or）

```
>>> 3 > 2 or 1 > 0 #True or True 会返回 True
True
>>> 3 > 2 or 1 < 0 #True or False 会返回 True
True
>>> 3 < 2 or 1 > 0 #False or True 会返回 True
```

```
True
>>> 3 < 2 or 1 < 0 #False or False 会返回 False
False
```

逻辑运算符“或”（or）的左右两边都是 False，才会返回 False，只要有任意一边是 True，就会返回 True。

3. 逻辑运算符——非（not）

```
>>> not 1 > 0 #not True 会返回 False
False
>>> not 1 < 0 #not False 会返回 True
True
```

逻辑运算符“非”（not）将会反转布尔值结果，“非真”会返回“假”，“非假”会返回“真”。

我们将“与”“或”“非”三个逻辑运算符的规律总结如下：

A	B	A and B	A or B	not A
True	True	True	True	False
True	False	False	True	
False	True	False	True	True
False	False	False	False	

了解了逻辑运算符的规律后，我们来尝试下面的几个练习吧！

逻辑运算符练习：

1. (True or False) and (False or True) 会返回?
2. 10 > 2 and (5 > 9 or 6 > 5) 会返回?
3. (True and 9 < 0) or 6 > 3 会返回?
4. 6 == 5 or 6 != 6 会返回?

4.6.5 案例 4：卡片通关挑战

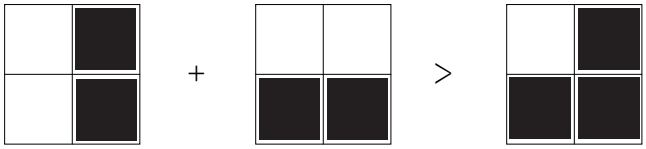
在前面我们学习了布尔值和逻辑运算符。在下面的练习里，我们不输入代码，而是换一个场景运用布尔值和逻辑运算符。

在这个练习里，我们将有两张卡片，卡片 A 和卡片 B 重叠，将得到卡片 C。重叠的规律将按

照逻辑运算符的规律，我们来看看吧。

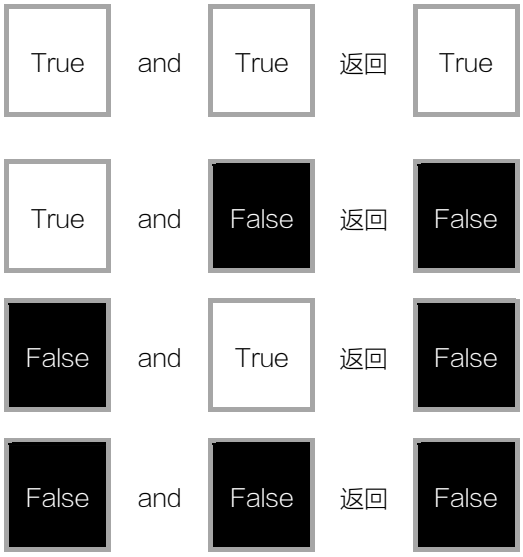
（1）与（and）的挑战

卡片 A 与卡片 B 叠起来会变成卡片 C。

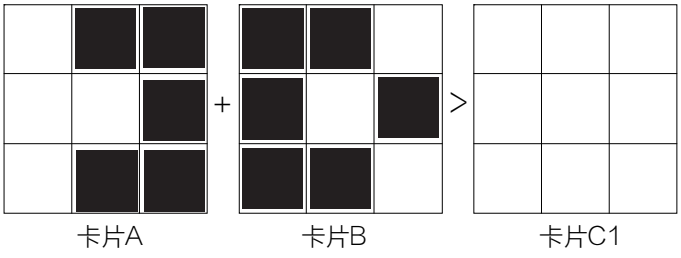


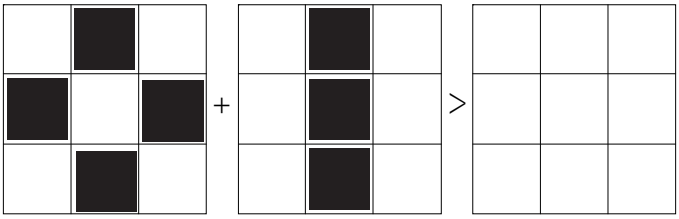
这个规律其实就是 and 逻辑运算符。

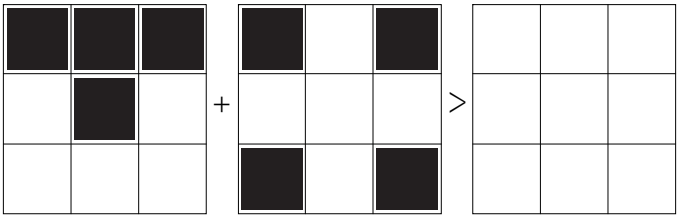
假设白色代表 True，黑色代表 False。



根据这个 and 规律，尝试一下下面的挑战，卡片 C1、C2、C3 是怎么样的呢?

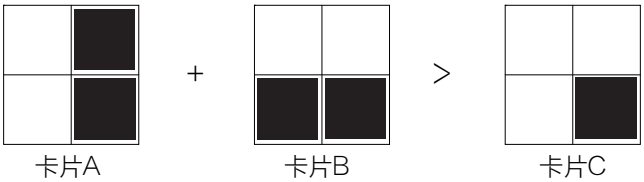






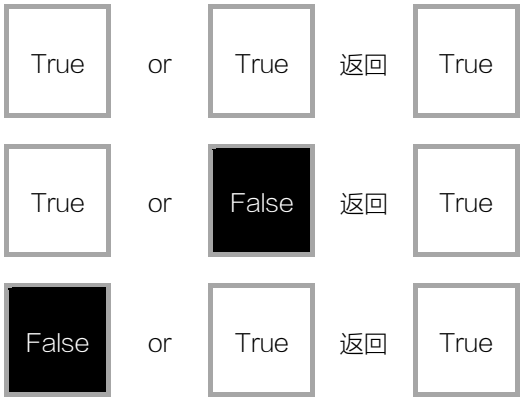
（2）或（or）的挑战

卡片 A 与卡片 B 叠起来会变成卡片 C。

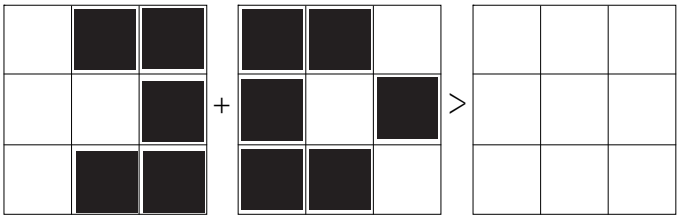


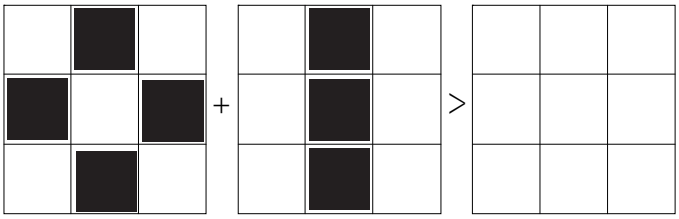
这个规律其实就是 or 逻辑运算符。

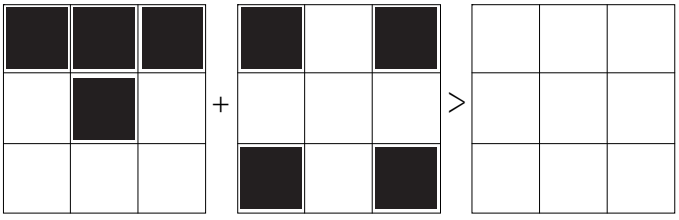
假设白色代表 True，黑色代表 False。



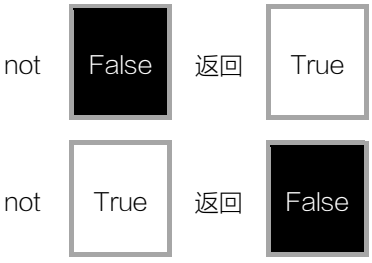
根据这个 or 规律，尝试一下下面的挑战，卡片 C4、C5、C6 是怎么样的呢?





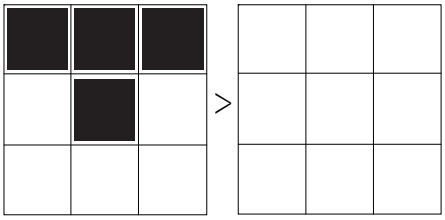


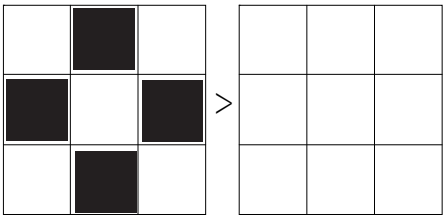
（3）非（not）的挑战

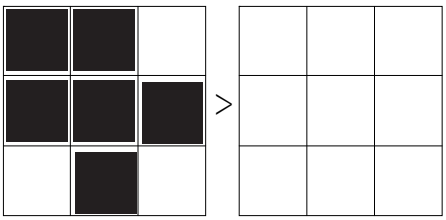


这个规律是 not 逻辑运算符。

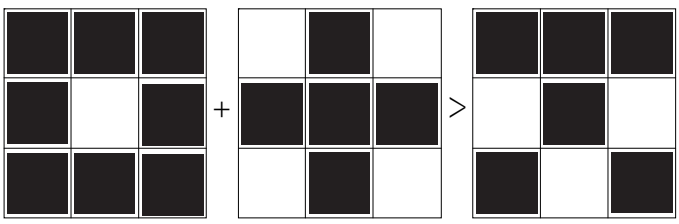
根据这个 not 规律，尝试一下下面的挑战，卡片 C7、C8、C9 是怎么样的呢?

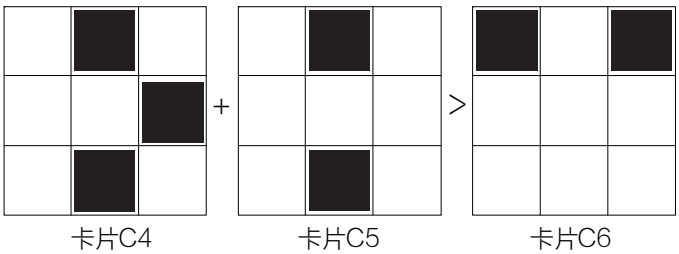


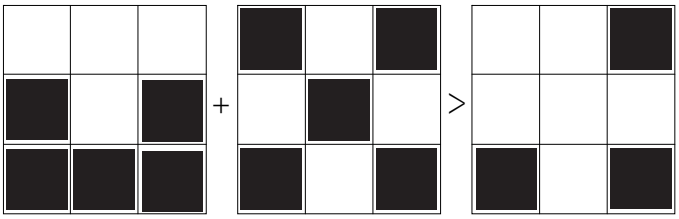




答案：







4.7　总结及课后练习

在这一章里，我们了解了变量是什么，并且学习了数字、字符串、布尔值三种数据类型。

1. 8 % 5 + 3 的结果是_____。

2. 420 // 100 * 2 的结果是_____。

3. 下面代码打印出的结果是什么？

a = 35 a += 1 a /= 2 print(a)	a = "12.93" a = float(a) a = int(a+1) print(a)

4. 下面代码打印的结果是什么？a =“世上无难事，只怕有心人”。

b = a * 3 print(b)	b = a[-2] print(b)	b = a[3] + a[7] print(b)
b = a[len[a]- 1] print(b)	a = a.replace ("怕有心人", "要肯放弃") print(a)	

5. 下面代码将会返回什么值？

12 > 5　　返回_____

12 > 5 and 7 != 8　　返回_____

(True or False) and 8 == 8　　返回_____

第 5 章

好好安排数据

本章重点是让读者思考信息世界的数据：为什么要有不同的数据结构，了解如何用数据结构将现实生活中复杂的情况简化并进行抽象表达，方便解决问题。

5.1 安排数据的方式

上一章我们介绍了常用的数据类型，包括数字、字符串等。这一章我们将介绍更多的数据类型，这些数据类型有助于我们更好地安排数据，方便我们表达数据、寻找数据和处理数据。

例如，要编写一个井字游戏，游戏棋盘如图 5-1 所示。若要在计算机里保存棋盘盘面，有几种选择？比如可以将它作为一张图片保存，但图片很占空间，最终效果不理想。为了节省空间，方便存储，于是决定寻找更好的方法。

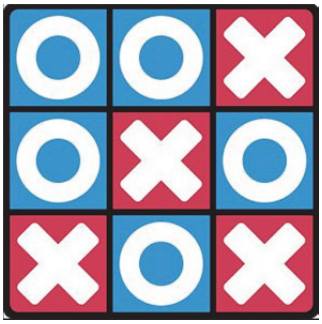


图 5-1　井字游戏

思考后发现可以用已知的数据类型代替圆圈、叉号。例如：

- 用字符串"O"代表圆圈
- 用字符串"X"代表叉号

从上往下数可以用"OOXOXOXOX"代表棋盘，只要存储这一串字符，就可以知道棋盘盘面长什么样了。除 O 和 X 外，还可以用数字代表圆圈和叉号。

- 用 0 代表圆圈
- 用 1 代表叉号

从上往下数可以用"001010101"代表这个棋盘，只要存储这一串数字，就可以知道棋盘盘面长什么样了。或者可以用一个列表代表它[0, 0, 1, 0, 1, 0, 1, 0, 1]。但若想找到第二行的所有数字，就要一个个地数，找索引值 1、4、7 上的数字不太方便。其实可以每一列用一个列表表示：[[0, 0, 1], [0, 1, 0], [1, 0, 1]]，这样可以更明显地表示出不同行、不同列之间的关系。

在这一章里，我们将学习安排数据的不同方式。列表是其中一种，本章还会介绍元组和字典。学习之后，我们就能根据不同的需求选择合适的方式安排数据。

5.2　列表

在生活中，常出现不同的列表，如图 5-2 所示。这些列表将一系列的数据收集起来，方便我们查找。

图 5-2　生活中的列表

在 Python 里，列表（list）用方括号表示，列表里的数据可以是任何数据类型。我们可以直接用等号“=”创建列表，给变量赋值。

```
friends = ["刘骏", "张飞", "芭娜娜魔女"]
tasks = ["交作业", "给朋友买礼物", "取快递", "买牛奶", "计划十一活动"]
scores = [96, 95, 99]
books = ["《围城》", "《安娜 · 卡列尼娜》", "《伟大的盖茨比》", "《活着》"]
things = [23, "牧羊犬", [1, 3, 4]]
newList = []
```

变量 friends 里存储着朋友名单，每个人名都是字符串；变量 tasks 里存储着待办事项列表，每个事项都是字符串；变量 scores 里存储着三门课的分数，每个分数都是整数；变量

books 里存储着四本书列表，每个书名都是字符串；变量 things 里存储的列表里有好几种不同的数据类型，包括数字、字符串、列表；而变量 newList 里存储着一个空的列表。

5.2.1 获取列表值

学会创建列表后，我们就要学会使用列表。首先，我们来看看如何读取列表里存储的列表项。每个列表项都有它的索引值，指出它在列表中的位置。跟字符串一样，列表的索引值从 0 开始。

```
tasks = ["交作业", "给朋友买礼物", "取快递", "买牛奶", "计划十一活动"]
```

列表项	"交作业"	"给朋友买礼物"	"取快递"	"买牛奶"	"计划十一活动"
索引值	0	1	2	3	4

如果你想获取单个列表项，则需要用索引值找到它。例如：

```
>>>tasks = ["交作业", "给朋友买礼物", "取快递", "买牛奶", "计划十一活动"]
>>>tasks[0]
"交作业"
```

```
>>>tasks[1]
"给朋友买礼物"
```

```
>>>tasks[4]
"计划十一活动"
```

```
>>>tasks[5]
IndexError: list index out of range
#错误信息！没有索引值为 5 的列表项
```

打开 IDLE，添加下面 4 个列表。

```
>>>friends = ["刘骏", "张飞", "芭娜娜魔女"]
>>>tasks = ["交作业", "给朋友买礼物", "取快递", "买牛奶", "计划十一活动"]
>>>scores = [96, 95, 99]
>>>books = ["《围城》", "《安娜 · 卡列尼娜》", "《伟大的盖茨比》", "《活着》"]
```

尝试做一下下面几个练习，看看打印出什么内容？

列表练习 1

```
a = friends[1]
print(a)
```

列表练习 2

```
b = 0
c = scores[b]
print(c)
```

列表练习 3

```
d = friends[2] + "要" + tasks[2]
print(d)
```

列表练习 1

1. 设变量 a 为 friends 列表里索引值为 1 的项，字符串为“张飞”
2. 打印变量 a 里存储的值
   结果是张飞

```
>>>a = friends[1]
>>>print(a)
张飞
```

列表练习 2

1. 设变量 b 为 0
2. 提取变量 b 里的值 0，找到 scores 列表中索引值为 0 的项，即数字 96，存进变量 c 里
3. 打印变量 c 里存储的值
   结果是 96

```
>>>b = 0
>>>c = scores[b]
>>>print(c)
96
```

列表练习 3

1. 找到 friends 中索引值为 2 的项“芭娜娜魔女”和 tasks 中索引值为 2 的项“取快递”，用这两个字符串和“要”一起组成一个大字符串，存进变量 d 里
2. 打印变量 d
   结果是芭娜娜魔女要取快递

```
>>>d = friends[2] + "要" + tasks[2]
>>>print(d)
芭娜娜魔女要取快递
```

通过索引值，我们能够找到列表项，并进一步对它们进行处理。

如询问一个列表项是否在列表中，可使用关键字 in，会返回相应的布尔值：

```
>>>books = ["《围城》", "《安娜·卡列尼娜》", "《伟大的盖茨比》", "《活着》"]
>>>"《围城》" in books
True

>>>"《红楼梦》" in books
False
```

字符串“《围城》”在 books 列表里，所以返回 True，而字符串“《红楼梦》”不在 books 列表里，所以返回 False。

如果要查找列表的长度，则可以使用 len()方法：

```
>>>books = ["《围城》", "《安娜·卡列尼娜》", "《伟大的盖茨比》", "《活着》"]
>>> len(books)
4
```

现在列表 books 里有 4 项元素，列表长度为 4，所以 len()函数返回 4。

我们可以观察到，列表中的许多处理方法其实与字符串类似。下一节将介绍如何修改列表的内容。

5.2.2 修改列表

有时候我们需要修改列表里的内容，包括修改列表项，给列表增加内容或者删除列表里的内容。

接着上面的例子，下面有一系列的任务要完成。

```
>>>tasks = ["交作业", "给朋友买礼物", "取快递", "买牛奶", "计划十一活动"]
```

1. 修改列表项

假设自己对奶糖过敏，不能买牛奶，只能买豆奶，可以通过索引值找到要修改的列表项，给它重新赋值：

```
>>>tasks = ["交作业", "给朋友买礼物", "取快递", "买牛奶", "计划十一活动"]
>>>tasks[3] = "买豆奶"
>>>tasks
```

```
["交作业", "给朋友买礼物", "取快递", "买豆奶", "计划十一活动"]
```

2. 添加列表项

需要添加一个“体检”任务进任务列表时，有以下两种方法。

方法 1：使用 append()方法在列表末尾添加项。

```
>>>tasks = ["交作业", "给朋友买礼物", "取快递", "买豆奶", "计划十一活动"]
>>>tasks.append("体检")
>>>tasks
["交作业", "给朋友买礼物", "取快递", "买豆奶", "计划十一活动", "体检"]
```

方法 2：用 insert()方法将新的列表项插入到列表特定的位置。例如体检是一项挺重要的事情，想要将它放到列表的第二位中，也就是索引值为 1 的地方。

```
>>>tasks = ["交作业", "给朋友买礼物", "取快递", "买豆奶", "计划十一活动"]
>>>tasks.insert(1, "体检")
>>>tasks
["交作业", "体检", "给朋友买礼物", "取快递", "买豆奶", "计划十一活动"]
```

3. 删除列表项

交完作业后，可以用 remove()方法将交作业从列表中删除。

```
>>>tasks
["交作业", "体检", "给朋友买礼物", "取快递", "买豆奶", "计划十一活动"]
>>>tasks.remove("交作业")
>>>tasks
["体检", "给朋友买礼物", "取快递", "买豆奶", "计划十一活动"]
```

当发现任务实在太多时，打算删掉一些任务，可以用 pop()方法删除。

```
>>>tasks
["体检", "给朋友买礼物", "取快递", "买豆奶", "计划十一活动"]
>>>tasks.pop(3) #删除索引值为 3 的列表项
"买豆奶"
>>>tasks
["体检", "给朋友买礼物", "取快递", "计划十一活动"]
>>>tasks.pop() #括号内若没有索引值参数，则删除最后一项
"计划十一活动"
>>>tasks
["体检", "给朋友买礼物", "取快递"]
```

4. 给列表排序

scores 列表有一系列的分数，若想让它们按从小到大的方式排序，可以用 sort()方法，也可以用 reverse()方法调转列表的顺序：

```
>>>scores = [96,95,99]
>>>scores.sort()
>>>scores
[95,96,99]
>>>scores.reverse()
>>>scores
[99,96,95]
```

列表的处理方法有很多，这里介绍了一些常用的方法，如果你有其他想要对列表进行的改动，则可以上网搜索，尝试查找。

5.2.3 二维列表

在前面的例子里，列表项主要是数字和字符串，但列表项还可以是其他数据类型，比方说列表里的列表项也可以是列表。

```
game1 = [[0,0,1],[0,1,0],[1,0,1]]
game2 = [[64,16,8,4],[16,4,2,0],[8,0,0,0],[2,0,0,0]]
```

以图 5-3 游戏棋盘为例，我们用列表来抽象化下面两个游戏的棋盘。左边的井字游戏棋盘用 game1 列表表示，右边的 2048 棋盘可以用 game2 列表表示。game2 列表有 4 个列表项，每个列表项自身又是一个列表，分别代表棋盘上的 4 行。

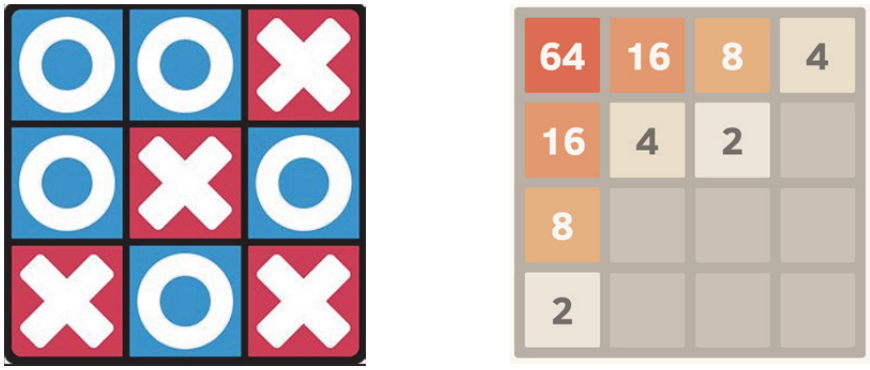


图 5-3 左为井字游戏棋盘，右为 2048 棋盘

game2[1][2]将返回第 2 行第 3 列上的数字 2。

```
>>>game2 = [[64,16,8,4],[16,4,2,0],[8,0,0,0],[2,0,0,0]]
>>>game2[1] #1 为索引值，返回 game2 里索引值为 1 的列表
[16,4,2,0]
>>>game2 = [[64,16,8,4],[16,4,2,0],[8,0,0,0],[2,0,0,0]]
>>>game2[1][2] #返回 game2 里索引值为 1 的列表里索引值为 2 的列表项
2
```

若要改变棋盘现状，例如，右下角出现了一个新的 2，如图 5-4 所示。可以用 game2[3][3] = 2 修改列表，得到更新后的 game2 棋盘。

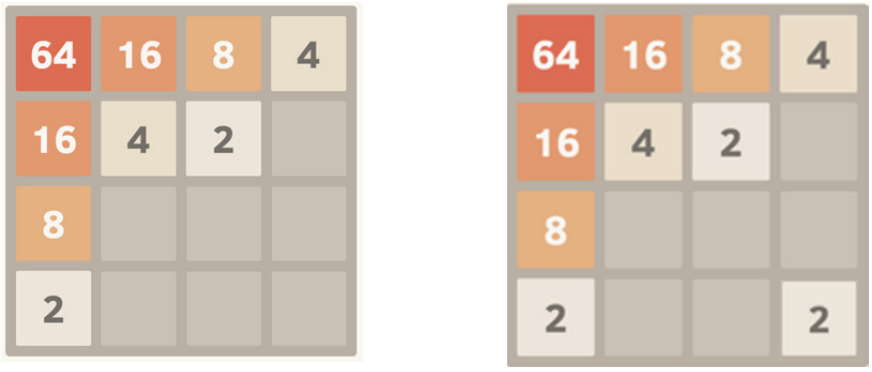
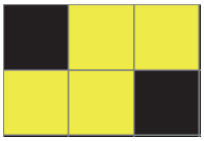
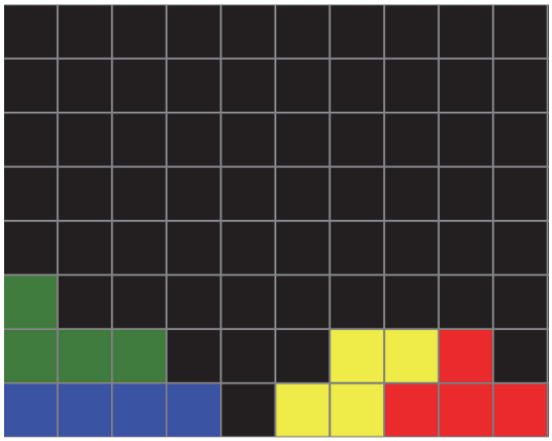


图 5-4　2048 棋盘右下角出现数字 2

```
>>>game2 = [[64,16,8,4],[16,4,2,0],[8,0,0,0],[2,0,0,0]]
>>>game2[3][3] = 2
>>>print(game2)
  [[64,16,8,4],[16,4,2,0],[8,0,0,0],[2,0,0,2]]
```

在《俄罗斯方块》这款游戏里，我们也可以用列表表示不同的形状和游戏画面。

[[1, 0, 0], [1,1,1]]	
[[0, 2, 0], [2, 2, 2]]	

[[0, 3, 3], [3, 3, 0]]	
[4, 4, 4, 4]	
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0, 0, 0, 0, 0], [1, 1, 1, 0, 0, 0, 3, 3, 2, 0], [4, 4, 4, 4, 0, 3, 3, 2, 2, 2]]	

了解列表的灵活应用后，我们可以更好地抽象化任务，方便编程。

所谓“抽象化”，就是省略复杂的细节，把最重要的信息用简单的符号表示出来。比方说棋盘盘面可能很复杂，有各种不同的颜色、形状、数字等。我们忽略这些复杂的细节，把核心的颜色用简单的数字代替，把列和行用列表表达出来，这种简化问题的方式就是“抽象”。我们可以用二维列表很好地抽象出游戏棋盘。

了解了不同的列表后，我们来做一些练习，之后去看看其他的数据类型。

5.2.4 列表挑战练习

1. 下面的代码会打印出什么？

```
songs = ["布拉格广场", "Champion", "我想起你了"]
songs.append("黑色幽默")
print(songs)
```

2. 下面的代码会打印出什么？

```
songs = ["布拉格广场", "Champion", "我想起你了"]
print(songs[2] + songs[0])
```

3. 下面的代码会打印出什么？

```
songs = ["布拉格广场", "Champion", "我想起你了"]
songs.pop()
print(songs)
```

4. 尝试抽象化图 5-5 所示的图片，用 0 代表白色，1 代表黑色。

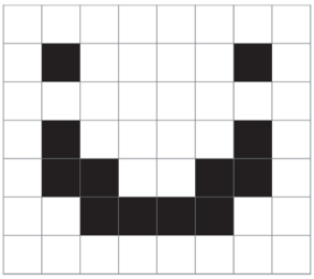


图 5-5　黑白像素图片

5.3　元组

元组（tuple）和列表（list）非常类似，同是一串数据，可以存储任何数据类型，用整数当索引值。

```
#列表使用方括号
myList = [97,91,93,96,99]

#元组使用圆括号
myTuple = (2018,12,31,23,59)
```

在语法上，列表用的是方括号，而元组用的是圆括号。它们的主要区别在于元组一旦生成，就不能被改变。我们来练习下面两段代码。

```
>>> myList = [97,91,93,96,99]
>>> myList[2] = 0
>>> print(myList)
[97,91,0,96,99]
```

```
>>> myTuple = (2018,12,31,23,59)
>>> myTuple[2] = 0
TypeError: 'tuple' object does not support item
assignment
```

可以看到，我们能够将列表 myList 索引值为 2 的列表项修改成 0，但如果在元组上尝试同样的操作，将元组 myTuple 索引值为 2 的项改为 0，则会报错，提示不能重新赋值。

很多时候，列表会用来表示一连串意义相同的数据，例如，myList 里 97、91、93、96、99 可能是 5 个不同的分数，是 5 项单独的数据。而元组则用来表示一系列不同意义的数据，一串数据会被当成整体来看。例如，myTuple 里 2018、12、31、23、59 一整串数字可能代表了时间点 2018 年 12 月 31 日 23 时 59 分。这个元组里的每一项意义都不同，有的代表年份，有的代表月份等，但它们组成了一个整体。

5.4 字典

5.4.1 什么是字典

前面的列表和元组都收集一连串的数据。而数据有时候是成对出现的，比方说英汉字典中的条目、电话簿中的记录等。

Apple	苹果
Banana	香蕉
Candy	糖果
Dog	狗
Egg	鸡蛋
Father	父亲

小赵	13066666666
小钱	15012345678
小孙	18900000000

针对这种特性，Python 提供了字典（dictionary）数据类型。

我们来创建一个字典，存在 eng2chi 变量里，有 6 项数据，存储 6 个单词：

```
eng2chi = {"Apple":"苹果", "Banana":"香蕉", "Candy":"糖果", "Dog":"狗" , "Egg":"鸡蛋" , "Father":"父亲"}
```

我们再创建一个字典，存在 phone 变量里，这个字典有 3 项数据，存储 3 个同学的电话号码。

```
phone = {"小赵":13066666666, "小钱":15012345678, "小孙":18900000000}
```

字典的功能和前面的列表与元组类似，都是收集一系列的数据。但列表和元组使用整数为索引值，而字典的索引值则可以是多种不同的数据，我们称它们为键（key）。

在 phone 这个字典里，"小赵"、"小钱"、"小孙"是键（key），它们对应的 13066666666、15012345678、18900000000 是值（value），每一对键和值组成一对数据。

在 eng2chi 这个字典里，"Apple"、"Banana"、"Candy"等是键（key），它们对应的"苹果"、"香蕉"、"糖果"等是值（value）。

5.4.2　使用字典

跟之前的列表不同，要想读取数据，就需要使用键，以此读取它对应的值。例如，如果想找到小赵的电话号码，就需要用 phone["小赵"]找到 13066666666。

```
>>> phone = {"小赵":13066666666, "小钱":15012345678, "小孙":18900000000}

>>> print(phone["小赵"])
13066666666

>>> print(phone["小孙"])
18900000000

>>> print(phone[0])
KeyError: 0
#0 不能为索引值，所以报错
```

如要修改数据，可以通过键找到要修改的数据并把新的值赋值给它。

```
>>> phone = {"小赵":13066666666, "小钱":15012345678, "小孙":18900000000}
>>> phone["小赵"] = 13000000000
>>> print(phone)
{"小赵": 13000000000, "小钱": 15012345678, "小孙": 18900000000}
```

如要往字典里添加数据，只要在方括号里使用新的键并设好它的值，就可以将新的数据加进来了。如下：

```
>>> phone = {"小赵":13066666666, "小钱":15012345678, "小孙":18900000000}
>>> phone["小李"] = 13898765432
>>> print(phone)
{"小赵": 13066666666, "小钱": 15012345678, "小孙": 18900000000, "小李": 13898765432}
```

如要删除某个条目，使用 del 关键词，找到键就可以删除了。如下：

```
>>> phone = {"小赵":13066666666, "小钱":15012345678, "小孙":18900000000}
>>> del phone["小赵"]
```

```
>>> print(phone)
{"小钱": 15012345678, "小孙": 18900000000}
```

5.4.3 案例：查询课程表

任务：下面有一个课程表，请创建一个字典，以“时间:课程”的方式呈现下面的课程表或者你自己的课程表。编写程序让用户通过输入星期几来查询相应的课程。

星期一	星期二	星期三	星期四	星期五
语文	化学	物理	数学	英语
数学	英语	音乐	语文	语文
编程	体育	历史	地理	数学
英语	生物	美术	生物	体育

实例

你要查星期几的课程表：星期一
语文，数学，编程，英语

timetable1["星期一"]将返回"语文，数学，编程，英语"；

timetable1["星期二"]将返回"化学，英语，体育，生物"；

timetable1["星期三"]将返回"物理，音乐，历史，美术"。

自己尝试完成这个任务吧！如果不会，再跟我一起打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好后继续编写。

查询课程表 1

1. 创建一个字典，存进 timetable1 变量中，星期几和课程表均为字符串
2. 用 input()让用户输入星期几存进变量 day 中
3. 以 day 为键，找到相应的课程表字符串

```
1. timetable1 = {"星期一":"语文，数学，编程，英语",
                 "星期二":"化学，英语，体育，生物",
                 "星期三":"物理，音乐，历史，美术",
                 "星期四":"数学，语文，地理，生物",
                 "星期五":"英语，语文，数学，体育"}

2. day = input("你要查星期几的课程表：")
3. print(timetable1[day])
```

点击【Run】→【Run Module】执行程序。

根据我们编写的程序，用户输入的信息会存在变量 day 里，我们通过 day 里面的星期字符串为键找到相应的课程表字符串，并打印出来。

我们来尝试改进代码，让用户可以具体查第几节课。

```
你要查星期几的课程表：星期一
你要查第几节课？请给出数字：2
数学
```

在编写程序之前，我们先退一步想想。如果课程表字典里的值"语文，数学，编程，英语"用字符串表示，那么我们并不容易区分出第 1 节、第 2 节是什么。但是如果每个值都是一个列表，例如，["语文", "数学", "编程", "英语"]就容易得多，我们可以根据索引值找到查询的课。

另存一份 Python 文件（Shell 窗口：【File】→【Save As...】），在之前的基础上进行如下修改。

查询课程表 2

1. 创建一个字典，存进 timetable2 变量中，星期几为字符串，课程为列表，每节课是一个字符串
2. 用 input()让用户输入星期几存进变量 day 中
3. 用 input()让用户输入想查询的第几节课，用 int()将它转换成数字，存入变量 lesson 中
4. 以 day 为键，找到相应的课程列表，再以 lesson-1 为索引值，找到对应的课程字符串，打印出来

```
1. timetable2 = {"星期一":["语文", "数学", "编程", "英语"],
                 "星期二":["化学", "英语", "体育", "生物"],
                 "星期三":["物理", "音乐", "历史", "美术"],
                 "星期四":["数学", "语文", "地理", "生物"],
                 "星期五":["英语", "语文", "数学", "体育"]}

2. day = input("你要查星期几的课程表:")
3. lesson = int(input("你要查第几节课？请给出数字："))
4. print(timetable2[day][lesson - 1])
```

点击【Run】→【Run Module】执行程序。

其实仔细想想，第二种方法要输入两个信息，用户用起来可能还不如第一种方法方便，不过就当是练习字典和列表吧。在今后的程序编写中，也不要忘了从用户角度想一想，这是否是最优的解决方案？

5.4.4 字典挑战练习

下面有一个字典，存在变量 song 里。

```
song = {
   "id": 133534636,
   "名称":"匈牙利广场",
   "歌手": "蔡小林",
   "语言": "中文",
   "风格": ["流行", "摇滚", "热门"],
   "排名": 5
}
```

1. 执行完下列两行代码后，这首歌的数据将有什么改变？

```
song["风格"].remove("流行")
song["风格"].append("怀旧")
```

2. 执行完下列代码后，这首歌的数据将有什么改变？

```
song["排名"] += 100
```

3. 执行完下列代码后，这首歌的数据将有什么改变？

```
song["发行年份"] = 2003
```

尝试将代码编写出来，看看是否跟你的答案一致。

5.5 总结及课后练习

这一章我们了解了列表、元组和字典，灵活使用这些数据类型有助于我们为不同的编程任务设计最合适的数据类型。

1. 下面这段代码会打印出什么？

```
luckyNumbers = [2,3,5,12,15]
a = 2
a += 1
print(luckyNumbers[a])
```

2. 下面这段代码会打印出什么？

```
colors = ["red", "orange", "yellow", "green", "cyan", "blue", "purple"]
a = 9
```

```
print(colors[a % 7])
```

3. 根据下面的代码编写程序，让用户输入一个时间，查出它对应的时辰。

```
times = {23: "子", 24: "子", 1: "丑", 2: "丑", 3: "寅", 4: "寅", 5: "卯", 6: "卯", 7: "辰",
8: "辰", 9: "巳", 10: "巳", 11: "午", 12: "午", 13: "未", 14: "未", 15: "申", 16: "申", 17:
"酉", 18: "酉", 19: "戌", 20: "戌", 21: "亥", 22: "亥"}
```

现在是几点？ 12
12 点是午时

第 6 章

条件判断——学会做决定

本章重点是让读者了解条件判断，并体会条件判断在生活和程序中的应用。

6.1 条件判断

6.1.1 生活中的判断

在日常生活中，我们会做很多判断，根据不同的前提条件决定做不同的事情。看下面两个实例。

If the weather tomorrow is sunny, then I will go out and play football.

如果明天是晴天，那么我就出门踢足球。

If my score > 60, then I will go to the next grade, else I stay in the current grade.

如果我的分数大于 60，那么我将升入下一个年级，否则我将留级。

这是两个日常生活中我们做判断的例子。

6.1.2 程序中的判断

计算机程序也会做很多判断，根据不同的条件做不同的事情。比方说聊天软件会有登录页面，只有在密码输入正确时才能登录成功。我们的手机屏幕也会根据不同的条件做不同的事，如果电池电量足够，我会希望电池图标显示为绿色，而一旦电池电量比较少了，则希望电池图标显示为红色来提示用户，如图 6-1 所示。

If correctPassword == True, login, else display error message.

如果密码是正确的，就可以正常登录，否则显示警告信息。

If battery > 20%, then set battery color to green, else set battery color to red.

如果手机电量大于 20%，那么设电池图标颜色为绿色，否则设电池图标颜色为红色。

图 6-1　程序中的条件判断

在编写游戏时，希望能判断出游戏何时结束，比如：

If HP < 1, game over.

如果生命值小于 1，则游戏结束。

```
>>> HP = 30
>>> HP < 1
False
```

```
>>> HP = 0
>>> HP < 1
True
```

当 HP = 30 时，HP < 1 条件判断为 False（假），希望游戏继续；

当 HP = 0 时，HP < 1 条件判断为 True（真），希望游戏结束。

在上面的三个条件判断例子中，“如果”对应“if”，“否则”对应“else”。在 Python 语言里，我们使用 if 和 else 两个关键词编写基本的条件判断语句，让程序在不同的条件下做不同的事情。

6.2　if…else…语句

首先我们来学习 if…else…（如果……那么……）语句。

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），尝试输入下列代码，观察结果。

```
HP = 30
if HP < 1: # 30<1 为 False
→  print("游戏结束")
else:
```

```
HP = 0
if HP < 1: # 0<1 为 True
→  print("游戏结束")
else:
```

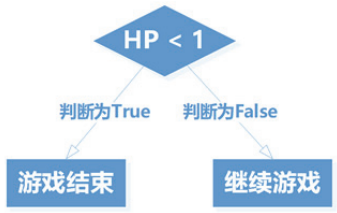
```
→ print("继续游戏")
#运行后打印"继续游戏"
```

```
→ print("继续游戏")
#运行后打印"游戏结束"
```

根据结果，在 if…else…语句中，如果条件为 True，则执行 if 冒号后的代码块；如果条件为 False，则执行 else 冒号后的代码块。

Python 中，if…else…语句的格式及含义如下：

```
if 条件:
    如果条件符合要做什么
else:
    如果条件不符合要做什么
```



6.2.1 案例 1：你的成绩合格吗

了解了 if…else…基本的语法后，我们来做个简单的练习：用 input()让用户输入成绩，判断这个数字是否大于或等于 60，并告诉用户是否合格。

实例

```
请输入你的分数>>>59.5
少年，回家继续努力吧
请输入你的分数>>>98
你合格了
```

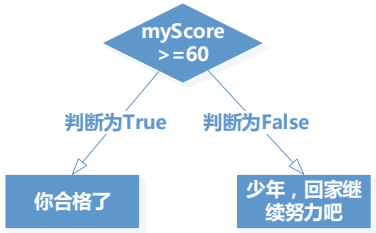
我们先把这个程序用 Python 语句的格式写出来，方便我们编写程序。

如果用户输入的数字大于或等于 60：

　　那么打印“你合格了”。

否则：

　　打印“少年，回家继续努力吧”。



请你尝试一下吧。值得注意的是，如果用户输入的数字是字符串，则需要调用 float()将字符串转换成浮点数。

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），并保存好。

你的成绩合格吗

1. 用 input()让用户输入分数，用 float()转换为浮点数并赋值给变量 myScore
2. 提取变量 myScore 的值，判断是否大于或等于 60
3. 如果判断为 True，打印"你合格了"
4. 否则
5. 打印"少年，回家继续努力吧"

```
1. myScore = float(input("请输入你的分数>>>"))
2. if myScore >= 60:
3. →  print("你合格了")
4. else:
5. →  print("少年，回家继续努力吧")
```

点击【Run】→【Run Module】执行程序，看看程序是否能够正确判断。

6.2.2　代码的位置

你可能注意到了，在输写代码的过程中，不是所有的代码都是顶格写的，if 和 else 冒号后的代码**缩进**了。缩进的意思是靠右一点。可以用单个制表符（按 Tab 键）或两个空格或四个空格进行缩进。在程序中，放在一起缩进的代码是一群**代码块**，从上到下执行。缩进的代码块隶属于它上面的代码。

小贴士：Python 缩进

Python 设计者是这么认为的：类似 C、Java 等语言用花括号{}来区分代码块。初学者通常不注意代码编写风格，它们认为花括号内是随便写的，是否分行，是否缩进都不影响代码逻辑。但这会破坏代码的层次结构，让人难以阅读。因此，许多有编程经验的人会不断给新手

们提示要注意缩进，让代码美观，避免给自己或他人造成阅读困难。

有人就问：何不把美观和逻辑结合在一起呢？终于，这套强制性的 Python 编写规则出来了，强制要求大家缩进，让代码美观易读。

6.2.3 案例 2：奇偶数判断

我们再做一个简单的条件判断练习：用 input()让用户输入一个数字，判断这个数字是奇数还是偶数。

实例

```
请输入数字>>>23
23 是奇数
请输入数字>>>20
20 是偶数
```

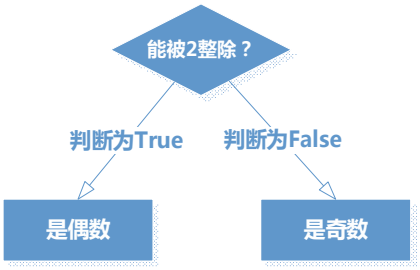
理解了题目要求后，我们用 Python 语句的格式写出来，方便编写程序。

如果用户输入的数字除 2 后余数是 0：

　　那么数字是偶数。

否则：

　　数字是奇数。



打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

到底是奇数还是偶数

1. 用 input()让用户输入数字，用 int()转换为整数并赋值给变量 myNum
2. 提取变量 myNum 的值求余 2，判断结果是否等于 0
3. 如果为 True，打印 myNum 是偶数
4. 否则
5. 打印 myNum 是奇数

```
1. myNum = int(input("请输入数字>>>"))
2. if myNum % 2 == 0:
3. →   print(str(myNum) + "是偶数")
4. else:
5. →   print(str(myNum) + "是奇数")
```

点击【Run】→【Run Module】执行程序，看看是否能够正确判断。

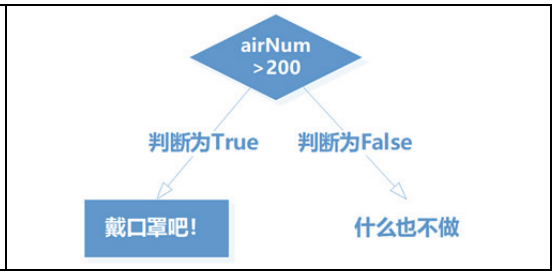
注意，这里用双等号“==”来比较左边和右边的值。单等号“=”是赋值符号，意思是设左边的变量为右边的值。

6.3　if…语句

掌握了 if…else…（如果……那么……）语句后，我们来学习 if…（如果……）语句。if…语句和 if…else…语句的不同之处在于前者如果不符合条件，就什么也不做，继续执行下面的程序，而后者会指出如果不符合条件时要做什么。

Python 中 if…语句示例如下。

```
if 条件：
    如果条件符合要做什么
```



案例 3：今天你戴口罩了吗

了解了 if…语句后，我们来练习这个例子：用 input()让用户输入雾霾颗粒指数（PM2.5 指数），如果这个指数大于 200，则告诉用户要戴口罩。

实例

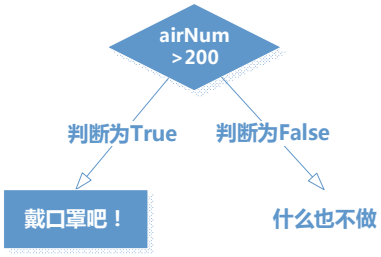
```
今天 PM2.5 指数多少>>>218.1
戴口罩吧!
今天 PM2.5 指数多少>>>188.5
```

我们把程序的要求用 Python 语句的格式写出来，方便编写程序。

如果用户输入的雾霾颗粒指数 ＞200：

　　那么打印“戴口罩吧!”

否则什么也不做。



请你尝试一下吧。打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

今天你戴口罩了吗

1. 用 input()让用户输入雾霾颗粒指数，用 float()转换为浮点数并赋值给变量 airNum
2. 提取变量 airNum 的值，判断是否大于 200
3. 如果为 True，打印“戴口罩吧!”

```
1. airNum = float(input("今天 PM2.5 指数多少>>>"))
2. if airNum > 200:
3. →  print("戴口罩吧！ ")
```

点击【Run】→【Run Module】执行程序。特别要确认如果条件不符合，是否就什么都不会出现呢?

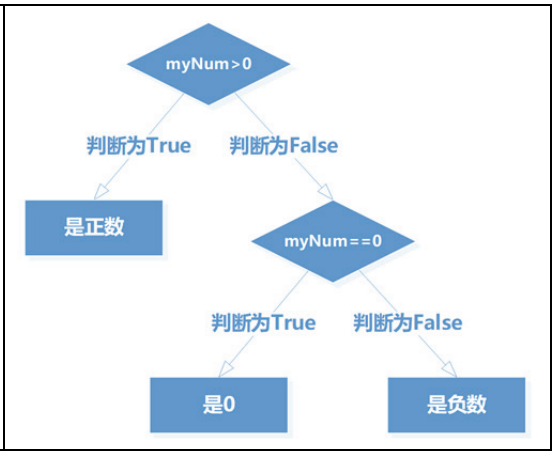
6.4　if…elif…else…语句

有时候我们有多重不同的条件，这就需要用到 if…elif…else…（如果……否则如果……否

则……）语句。

Python 中 if…elif…else…语句示例如下。

```
if 条件 1:
    如果条件 1 符合要做什么
elif 条件 2:
    如果不符合条件 1，但条件 2 符合要做什么
else:
    剩下的情况要做什么
```



我们可以根据需求添加多个 elif 条件，做多重条件判断。

案例 4：判断正数、负数和零

我们来看一个 if…elif…else…语句的例子：用 input()让用户输入一个数，判断这个数是正数、负数，还是 0。

实例

```
请给我一个数字>>>0
是 0
请给我一个数字>>>-1
是负数
请给我一个数字>>>1
是正数
```

我们先用 Python 语句理清程序的逻辑。

如果用户输入的数大于 0：

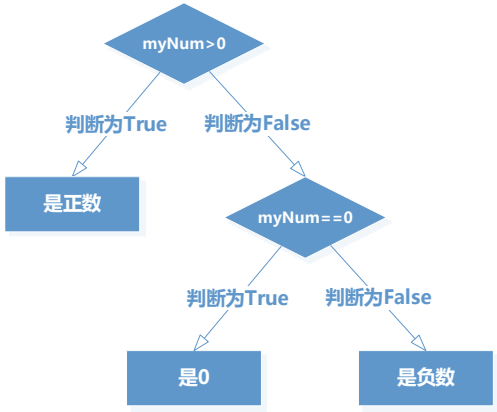
　　那么打印是正数。

否则，如果用户输入的数等于 0：

那么打印是 0。

否则：

打印是负数。



打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

判断正负数和零

1. 用 input()让用户输入一个数，用 float()转换为浮点数并赋值给变量 myNum
2. 提取变量 myNum 的值，判断是否大于 0
3. 如果为 True，打印"是正数"
4. 否则判断 myNum 的值是否等于 0
5. 如果为 True，打印"是 0"
6. 否则
7. 打印"是负数"

```
1. myNum = float(input("请给我一个数字>>>"))
2. if myNum > 0:
3. →  print("是正数")
4. elif myNum == 0:
5. →  print("是 0")
6. else:
7. →  print("是负数")
```

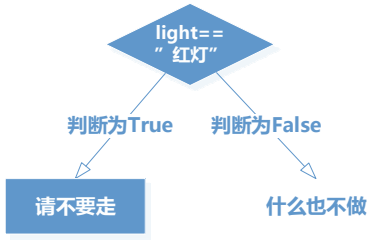
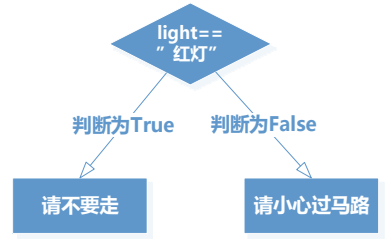
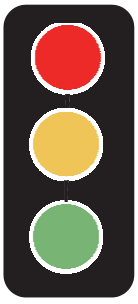
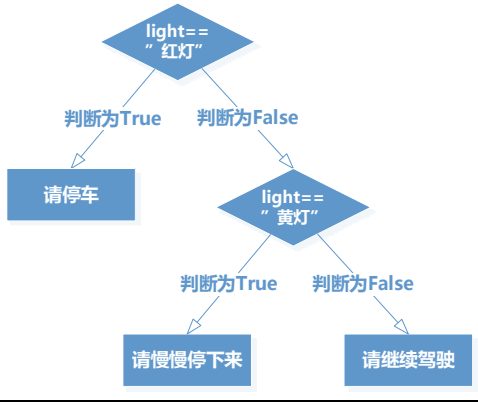
点击【Run】→【Run Module】执行程序。记住：至少要尝试三种不同的情况，以便确认一系列的条件判断都有效。

6.5　条件判断总结

我们通过下面的故事总结刚才提到的三种条件判断语句。

我们的小程序员为机器人王国设计了一套红绿灯导航系统，这套导航系统遵循“红灯停，绿灯行”的规则，会给行人和司机建议。下面根据规则，为行人和司机编写程序。

红绿灯导航系统设计：

红绿灯导航	设计要求	流程图
 行人导航 1	if 语句： 编写代码，让用户输入灯的状态，如果是红灯，打印“请不要走”	
 行人导航 2	If…else…语句： 我们发现很多行人绿灯时会看着手机过马路，这样非常危险，需要提醒行人“请小心过马路” 编写代码，让用户输入灯的状态，如果是红灯，打印“请不要走”，否则打印“请小心过马路”	
 司机导航	If…elif…else…语句： 编写代码，让用户输入灯的状态，如果是红灯，打印“请停车”，否则，如果是黄灯，打印“请慢慢停下来”，否则打印“请继续驾驶”	

6.5.1 红绿灯导航系统

下面尝试为上面三种不同的情况编写程序，分别用 if…语句、if…else 语句和 if…elif…else…语句。

程序运行效果	代码答案
请输入灯的状态>>>红灯 请不要走	light = input("请输入灯的状态>>>") if light == "红灯": → print("请不要走")

请输入是红灯还是绿灯>>>红灯
请不要走
请输入是红灯还是绿灯>>>绿灯
请小心过马路
	light = input("请输入是红灯还是绿灯>>>")
	if light == "红灯":
	→ print("请不要走")
	else:
	→ print("请小心过马路")

请输入红灯\绿灯\黄灯>>>红灯
请停车
请输入红灯\绿灯\黄灯>>>绿灯
请继续行驶
请输入红灯\绿灯\黄灯>>>黄灯
请慢慢停下来
	light = input("请输入红灯\绿灯\黄灯>>>")
	if light == "红灯":
	→ print("请停车")
	elif light == "黄灯":
	→ print("请慢慢停下来")
	else:
	→ print("请继续行驶")

通过上面的例子，我们可以清楚地看到这三种语句的不同。了解其特点后，我们在编程时就可以根据题目要求，使用合适的语句。来尝试一下下面的练习吧。

6.5.2 案例 5：闰年计算器

你知道平年和闰年的区别吗？一年通常有 365 天，是平年。而日历的年度天数实际上与地球公转周期有时间差。为了弥补这个时间差，产生了闰年（Leap Year）的概念，即这一年有 366 天，多出了 2 月 29 日。

下面编写一段程序，用 input()让用户输入年份，判断这个年份是否是闰年，并将结果打印出来。

实例

```
请输入年份>>>2000
2000 年是闰年
请输入年份>>>1900
1900 年不是闰年
```

```
请输入年份>>>1996
1996 年是闰年
请输入年份>>>1999
1999 年不是闰年
```

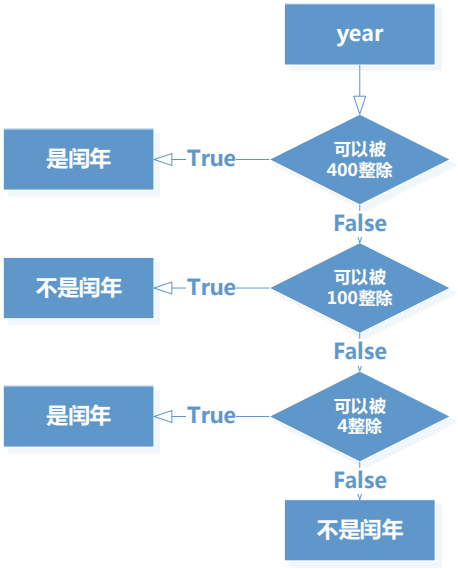
小贴士：闰年

普通年能被 4 整除且不能被 100 整除的为闰年。(如 2004 年就是闰年，2005 年不是闰年。)
世纪年要能被 400 整除才是闰年。(如 2000 年是闰年，1900 年不是闰年。)

如何编写这个程序呢?

我们来思考几种不同的情况。这几个条件中，最确定的是 400，如果能被 400 整除，就一定是闰年，这可以作为第一个条件。

如果不能被 400 整除，就要进一步看，判断是否能被 100 整除或是否能被 4 整除，这两个条件都能以不同的方法帮我们区分出闰年和平年。例如，先判断能否被 100 整除，如果判断为 True，我们就可以确认它不是闰年。如果判断为 False，接下来就用能否被 4 整除进行最后的区分。



在编写复杂的条件判断程序时，可以画出流程图，方便理清我们的思路。根据上面的流程图，我们可以编写出下面的代码，这里给出了两种不同的方法。

答案：闰年挑战

```
#解法 1
year = int(input("请输入年份>>>"))
if year % 400 == 0:
    print(str(year) + "年是闰年")
elif year % 100 == 0:
    print(str(year) + "年不是闰年")
elif year % 4 == 0:
    print(str(year) + "年是闰年")
else:
    print(str(year) + "年不是闰年")

#解法 2
year = int(input("请输入年份>>>"))
if year % 400 == 0 or (year % 100 != 0 and year % 4 == 0):
    print(str(year) + "年是闰年")
else:
    print(str(year) + "年不是闰年")
```

在解法 2 里，只用了一个条件，但是这个条件比较长，用 or 和 and 运算符，把几个判断串了起来。这个条件是它可以满足被 400 整除或者它同时满足不能被 100 整除与能被 4 整除。

编程挑战通常有许多不同的解法，希望大家多多尝试不同的语句，在探索中学会灵活使用编程语句。下面我们来看更多有趣的例子，练习灵活组合使用条件判断语句。

6.6 条件判断应用

6.6.1 案例 6：趣味掷骰子

掷骰子是一种随机游戏。假设周末我和爸、妈想要掷骰子决定去做什么，在下面的两个练习里，请根据条件判断语句的规则，给出 A、B、C、D 中可以达到目的的选项。

趣味掷骰子练习 1：

决定规则	下面哪一组骰子可以让我们去看电影

if 第 1 个骰子 > 第 2 个骰子:
　　待在家里
else:
　　if 第 3 个骰子 <第 2 个骰子:
　　去溜冰
	A 
	B 

续表

决定规则	下面哪一组骰子可以让我们去看电影
else: 　　去看电影	C  D 

趣味掷骰子练习 2：

决定规则	下面哪一组骰子可以让我们去溜冰

if 第 1 个骰子 < 第 2 个骰子:
　　待在家里
else:
　　if 第 3 个骰子 == 1:
　　　　去溜冰
　　else:
　　　　去看电影
	A 
	B 
	C 
	D 

参考答案见本书附赠资料。

6.6.2　案例 7：心理测验

你是否玩过互联网上的各种趣味测试?

你是哪个三国人物?

你适合在哪个城市居住?

穿越回古代你会是哪个皇帝?

测你的压力超标指数?

一秒测出你的多面性格?

……

其实，这些测试大多是程序员写出的搞笑的条件判断语句，首先设计几个问题，然后根据用户的回答给出不同的解释。我们学习了条件判断语句后，也可以设计和编写我们自己的趣味

测试。

让我们也来编写一个趣味测试，测试用户的心理年龄多少岁。下面编写 3 道题，给用户不同的选项，根据用户的答案，给它们设计出心理年龄。

实例

例 1

```
欢迎来到心理年龄测试!
第一题，朋友聚会，你喜欢去什么样的地方?
A－游乐场，B－唱 K，C－环境优美的餐厅。
请输入字母>>>B
第二题，你喜欢看哪种书?
A－《没头脑和不高兴》，B－《海边的卡夫卡》，C－《商战 101》，D－《道德经》。
请输入字母>>>B
第三题，你喜欢做哪种运动?
A－玩泥巴，B－运动是什么? C－跑步， D－太极拳。
请输入字母>>>C
你的心理年龄是 35 岁。
```

例 2

```
欢迎来到心理年龄测试!
第一题，朋友聚会，你喜欢去什么样的地方?
A－游乐场，B－唱 K，C－环境优美的餐厅。
请输入字母>>>A
第二题，你喜欢看哪种书?
A－《没头脑和不高兴》，B－《海边的卡夫卡》，C－《商战 101》，D－《道德经》。
请输入字母>>>A
第三题，你喜欢做哪种运动?
A－玩泥巴，B－运动是什么? C－跑步， D－太极拳。
请输入字母>>>A
你的心理年龄是 15 岁。
```

我们先设计这个趣味测验的问题、答案和分数。你也可以用你无限的创造力设计出其他有趣的问题和答案。

一开始假设 0 岁
欢迎来到心理年龄测试!
1. 朋友聚会，你喜欢去什么样的地方?
A. 游乐场 +5 岁
B. 唱 K +10 岁
C. 环境优美的餐厅 +15 岁

2. 你喜欢看哪种书?
A.《没头脑和不高兴》+5 岁
B.《海边的卡夫卡》 +10 岁
C.《商战 101》 +15 岁
D.《道德经》 +20 岁

3. 你喜欢做哪种运动?
A. 玩泥巴 +5 岁
B. 运动是什么? +10 岁
C. 跑步 +15 岁
D. 太极拳 +20 岁

设计好内容后，我们就准备编写程序了。这个程序看起来十分庞大，首先要学会将庞大的程序分解成小块，一段一段编写。我们可以把上面设计的心理测验分成以下几段:

段落	程序逻辑
开头	打印“欢迎来到心理年龄测试!” 设年龄 age 为 0
第一题	打印问题
第二题	打印选项
第三题	提示用户输入选项字母，存进变量中 用条件判断语句根据不同选项，加不同的心理年龄到 age 变量中
结果	打印出结果

这里的小技巧是先完成简单的段落，比方说先完成开头和结尾，再把第一题编写好。测验好第一题确认没有问题后，可以重复第一题的代码并加以修改，完成第二题和第三题。

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口:【File】→【New File】），保存好。

你的心理年龄是多少岁

```
#开头
print("欢迎来到心理年龄测试！")
age = 0

#第一题
print("第一题，朋友聚会，你喜欢去什么样的地方？")
print("A - 游乐场，B - 唱K，C - 环境优美的餐厅。")
q1 = input("请输入字母>>>")
if q1 == "A":
→  age += 5
elif q1 == "B":
→  age += 10
elif q1 == "C":
→  age += 15
else:
→  print("没看懂题？你0岁吗？")

#第二题
print("第二题，你喜欢看哪种书？")
print("A - 《没头脑和不高兴》，B - 《海边的卡夫卡》，C - 《商战101》，D - 《道德经》。")
q2 = input("请输入字母>>>")
if q2 == "A":
→  age += 5
elif q2 == "B":
→  age += 10
elif q2 == "C":
→  age += 15
elif q2 == "D":
→  age += 20
else:
→  print("没看懂题？你0岁吗？")

#第三题
print("第三题，你喜欢做哪种运动？")
print("A - 玩泥巴，B - 运动是什么？C - 跑步， D - 太极拳。")
q3 = input("请输入字母>>>")
if q3 == "A":
→  age += 5
elif q3 == "B":
```

```
→  age += 10
elif q3 == "C":
→  age += 15
elif q3 == "D":
→  age += 20
else:
→  print("没看懂题？你 0 岁吗？")

#结果
print("你的心理年龄是" + str(age) + "岁。")
```

点击【Run】→【Run Module】执行程序。

编写好后，我们可以思考如何改进这个程序。比方说用户输入了小写字母怎么办？（提示：可以用.upper()方法将用户的输入变成大写字母）找朋友一起测试一下程序，问问它们的意见，并对自己的程序加以修改。

6.6.3　案例 8：聊天机器人

聊天机器人（chatbot）是一个用来模拟人类对话或聊天的程序。手机里的聊天机器人可以当我的个人助理，告诉我现在的天气，给我解答简单的问题，帮我播放音乐，发短信、拨电话等。有的公司用聊天机器人为消费者提供咨询，有的玩具用聊天机器人给小朋友唱歌、讲故事等。

最早的聊天机器人叫 ELIZA，由美国麻省理工学院的 Joseph Weizenbaum 在 20 世纪 60 年代开发。许多人在和 ELIZA 聊天时，还以为自己在和真人讲话。

我们也可以设计一个简单的聊天机器人，使用一系列的条件判断语句，根据用户说话内容，给出相应的回答。我们可以从简单的程序做起，不断完善聊天机器人的功能。

先确定一个聊天场景。最好从具体的小场景开始设计。这里有几个想法，如下：

- 放学后聊天
- 聊电影
- 聊足球

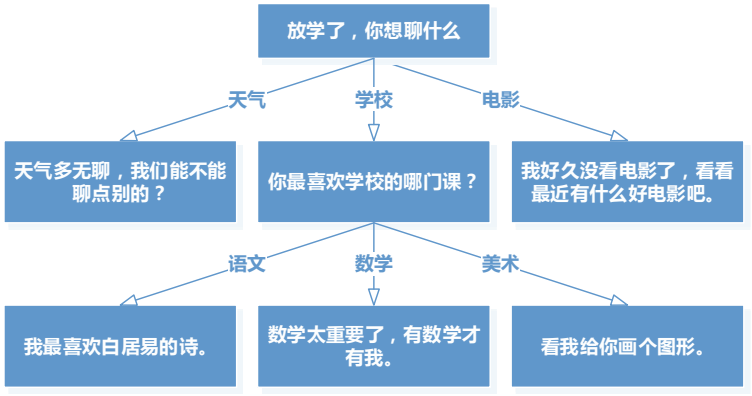
……

这里以“放学后聊天”为例进行进一步设计。选定场景后，我们会预判可能出现的关键词，并设计回复。

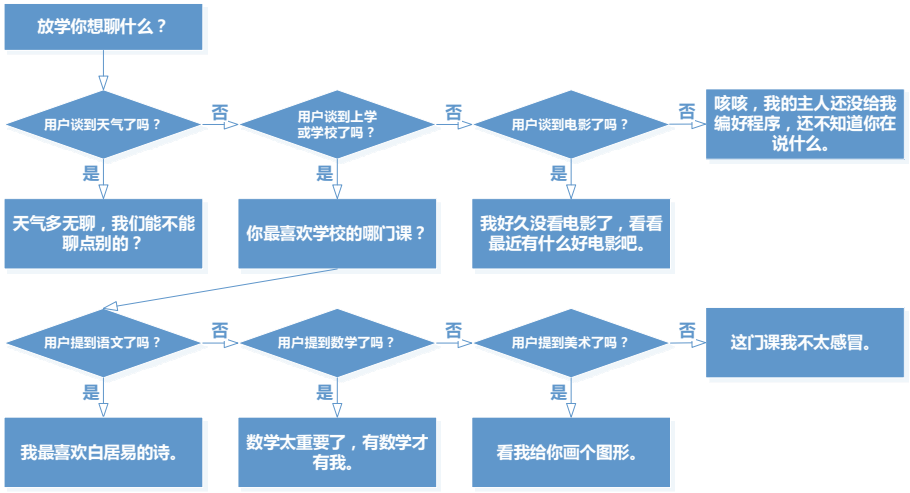
通常，学生放学后可能会出现“天气”“学校”“电影”等关键词，所以这里先以它们为例，设计回复。

关键词	回复
天气	“天气多无聊。我们能不能聊点别的？”
学校	“你最喜欢学校的哪门课?”
电影	“我好久没看电影了。看看最近有什么好电影吧。”

聊天机器人如果只给出一句回答难免有些无聊。下面以学校的学科为例，进一步延伸对话，问用户“你最喜欢学校的哪门课？”，并根据用户的回答进一步提问。



我们将用一系列的条件判断语句查看用户是否提到某些关键词，并根据关键词决定执行什么内容。



运行情况大致如下。

实例

例 1

放学了，你想聊什么>>>今天上学好累啊！
我最喜欢上学了！
你最喜欢学校的哪门课>>>语文吧。
我最喜欢白居易的诗。

例 2

放学了，你想聊什么>>>我刚刚踢足球回来。
咳咳，我的主人还没给我编好程序，还不知道你在说什么。

如果你觉得这个聊天机器人太简单了，不要着急。我们先把基础版本做好，之后可以根据基础版本发挥自己的想象，设计出有用且有趣的聊天机器人。

设计好内容后，我们就要开始编写程序了。跟之前一样，养成习惯将庞大的程序分解成小块，再逐步编写程序。

段落	程序逻辑
开头	询问用户“放学了，你想聊什么？”，将回答保存在变量 topic 里
第一种可能	如果 topic 里用户谈到了“天气”……
第二种可能	如果 topic 里用户谈到了“上学”或者“学校”……
第三种可能	如果 topic 里用户谈到了“电影”……
其他的谈话内容	如果上述条件都不符合，说“咳咳，我的主人还没给我编好程序，还不知道你在说什么。”

开头很简单，我们先编写开头程序。

开头	询问用户“放学了，你想聊什么？”，将回答保存在变量 topic 里

```
#开头
topic = input("放学了，你想聊什么>>>")
#第一种可能
#第二种可能
#第三种可能
#其他谈话内容
```

然后编写第一种可能，也是简单的条件判断语句，用 in 判断用户的回答 topic 里是否提到了

“天气”。

第一种可能	如果 topic 里用户谈到了“天气”，那么我打算让它说“天气多无聊。我们能不能聊点别的？”

```
#第一种可能
if "天气" in topic:
→ print("天气多无聊。我们能不能聊点别的？")
```

接下来再编写第二种可能。这里条件判断的大条件事实上是用两个小条件组成的，用 or 连接。根据 or 的规则，只要两边任一小条件判断为 True，大条件就是 True，符合我们想要的效果。

第二种可能	如果 topic 里用户谈到了“上学”或者“学校”，则进行进一步谈话。

```
#第二种可能
elif "学校" in topic or "上学" in topic:
#进一步对话
```

在这里先忽略进一步对话的代码，用注释来提示，之后再填充。忽略小细节，先把大框架搭起来，有助于我们有逻辑且快速地编写程序。

根据前面的规律，把第三种可能编写好。

第三种可能	如果 topic 里用户谈到了“电影”，则打印“我好久没看电影了。看看最近有什么好电影吧。”

```
#第三种可能
elif "电影" in topic:
→  print("我好久没看电影了。看看最近有什么好电影吧。")
```

最后，用 else 对一切其他的用户输入做回复。

其他的谈话内容	如果上述条件都不符合，则说“咳咳，我的主人还没给我编好程序，还不知道你在说什么。”

```
#其他谈话内容
else:
→  print("咳咳，我的主人还没给我编好程序，还不知道你在说什么。")
```

编写好后，我们的聊天机器人框架就出来了，可以完善第二种可能中标注的进一步的对话了。

段落	程序逻辑
开头	打印“我最喜欢上学了！”询问用户“你最喜欢学校的哪门课？”，将回答保存在变量 subject 里

续表

段落	程序逻辑
第一种可能	如果 subject 里用户谈到了“语文”，打印“我最喜欢白居易的诗。”
第二种可能	如果 subject 里用户谈到了“数学”，打印“数学太重要了，有数学才有我。”
第三种可能	如果 subject 里用户谈到了“美术”，打印“看我给你画个图形。”
其他的谈话内容	打印“这门课我不太感冒。”

```
#进一步对话
print("我最喜欢上学了！")
subject = input("你最喜欢学校的哪门课>>>")
if "语文" in subject:
    print("我最喜欢白居易的诗。")
elif "数学" in subject:
    print("数学太重要了，有数学才有我。")
elif "美术" in subject:
    print("看我给你画个图形。")
else:
    print("这门课我不太感冒。")
```

为了让我们的聊天机器人更加有趣，我们可以思考如何用已学的知识给它添加一些功能。

在谈美术时，不妨使用之前学过的 turtle 模块直接画一个图形。我们可以在代码前加上 import turtle，并打印“看我给你画个图形。”，然后添加画图形的程序。

```
import turtle

print("看我给你画个图形。")
t = turtle.Pen()
for i in range(120):
    t.forward(i)
    t.left(110)
```

在聊到电影时，我们不妨打开电影时间表，让用户可以看看最近有哪些上映的电影。

这里将用到暂时还没学过的 webbrowser 模块，我们在 turtle 后加上 webbrowser，导入 webbrowser 模块，并在讨论电影的环节加上 webbrowser.open()，打开电影时刻表的页面。这里用的是 http://theater.mtime.com/，你也可以换成其他查询电影的网页。

```
import turtle, webbrowser

#第三种可能
elif "电影" in topic:
```

```
    print("我好久没看电影了。看看最近有什么好电影吧。")
    webbrowser.open("http://theater.mtime.com/")
```

至此，这个案例的基础版本就完成了，下面是完整代码。

聊天机器人

```
import turtle, webbrowser

#开头
topic = input("放学了，你想聊什么>>>")

#第一种可能
if "天气" in topic:
    print("天气多无聊。我们能不能聊点别的？")

#第二种可能
elif "学校" in topic or "上学" in topic:
    #进一步对话
    print("我最喜欢上学了！")
    subject = input("你最喜欢学校的哪门课>>>")
    if "语文" in subject:
        print("我最喜欢白居易的诗。")
    elif "数学" in subject:
        print("数学太重要了，有数学才有我。")
    elif "美术" in subject:
        print("看我给你画个图形。")
        t = turtle.Pen()
        for i in range(120):
            t.forward(i)
            t.left(110)
    else:
        print("这门课我不太感冒。")

#第三种可能
elif "电影" in topic:
    print("我好久没看电影了。看看最近有什么好电影吧。")
    webbrowser.open("http://theater.mtime.com/")

#其他谈话内容
else:
    print("咳咳，我的主人还没给我编好程序，还不知道你在说什么。")
```

点击【Run】→【Run Module】执行程序，测试是否能够正确地运行。

我们还可以通过以下方法改进聊天机器人。

- 增加对话等级，接着深入聊语文、数学、美术。
- 增加对话质量，多给一些类似图画或链接的高质量内容。也可以把上一章做的课程表查询功能添加进聊天机器人里。
- 增加对话广度，除学校、天气、电影外，加入其他类别的聊天内容。

小贴士：聊天机器人

聊天机器人，简单地说，就是基于人工智能（Artificial Intelligence，以下简称 AI）原理，通过对聊天文本进行分析后给出应答的一类程序。下面介绍几种其他聊天机器人程序，大家可以参考作为拓展阅读。

1. 人人网上 AI 聊天机器人小黄鸡

   AI 聊天机器人小黄鸡的工作可以被分成两部分：训练+匹配。

   具体可参考：http://blog.csdn.net/huyisu/article/details/35552827

2. 用 Python AIML 搭建聊天机器人

   Python 需要安装 Python aiml 库，获取 Alice 资源，并加载 Alice 后，可与 Alice 聊天。

   具体可参考：https://www.biaodianfu.com/python-aiml.html

6.7　总结及课后练习

这一章我们了解了条件判断语句，利用它我们可以根据条件执行不同的代码块。

1. 运行下面的代码，将会打印出什么呢?

```
a = 36
b = 55
if a < 30:
    print('111')
if a < 20 or b > 13:
    print('222')
if a > 23 and a <= 40:
    print('333')
```

```
if b > 30:
    if b < 50:
        print('444')
    else:
        print('555')
```

一步一步判断每个 if 语句的结果是 True 还是 False。

2．这里有一系列的条件判断语句:

```
if x > 0:
    if y > 0:
        print("红色")
    else:
        if z > 0:
            print("蓝色")
        else:
            print("绿色")
else:
    print("粉红色")
```

下面几种条件分别会打印出什么结果?

A. x = −1, y = −1, z = −1

B. x = 1, y = 1, z = 0

C. x = 1, y = −1, z = 1

D. x = 1, y = −1, z = −1

把 x，y，z 代入程序中尝试。

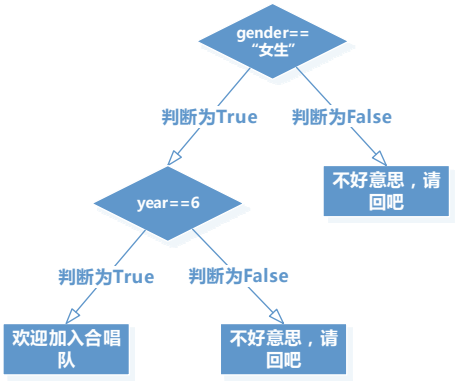
3．学校希望一位 6 年级的女生加入合唱队。请编写一段代码，询问学生的性别和就读年级，根据条件打印出是否可以加入合唱队。

例 1

```
请问你是男生还是女生>>>女生
请问你上几年级? 输入数字>>>6
欢迎加入合唱队
```

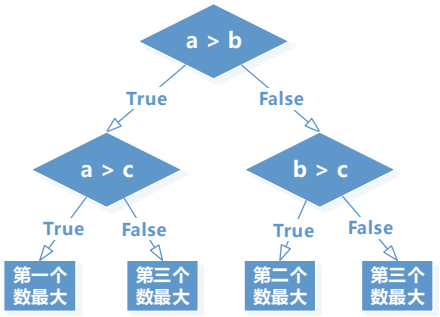
例 2

```
请问你是男生还是女生>>>男生
不好意思，请回吧
```



4．请编写一段代码，要求用户输入三个数字，比较三个数字的大小，并打印出第几个数字最大。

```
输入第一个数字>>>18
输入第二个数字>>>12
输入第三个数字>>>23
第三个数最大
```



第 7 章

循环——让计算机重复工作

本章重点是让读者了解循环并体会循环在程序中的应用。计算机的一大优点就是能通过循环重复地完成工作。

7.1 流程控制

在前面的章节中，我们了解到代码通常是从上往下按顺序执行的，一旦遇到条件判断语句，则要依情况选择执行相关的代码块。在这一章里，我们将了解循环语句，用循环语句重复执行某些代码块。Python 语言使用这些不同的语句控制如何执行代码，我们称为“流程控制”。

普通语句从上往下按顺序执行代码。

1. 往前走 10 步 2. 右转 90 度 3. 往前走 10 步 4. 往前走 10 步	1、2、3、4 每个指令执行 1 遍，从上往下按顺序执行

条件判断语句根据特殊条件，决定是否执行某些代码。

1. 往前走 10 步 如果右边有路: 　　2. 右转 90 度 　　3. 往前走 10 步 4. 往前走 10 步	先执行指令 1，只有在“右边有路”的条件满足的情况下，才执行 2、3 指令，最后执行指令 4

循环语句重复执行某些代码。

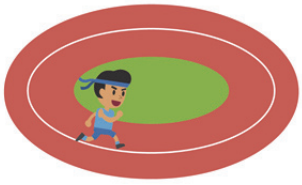
1. 往前走 10 步: 重复执行下面的代码 8 次: 　　2. 右转 90 度 　　3. 往前走 10 步 4. 往前走 10 步	先执行指令 1，然后重复执行指令 2 和 3 共 8 次，最后执行指令 4

7.2　什么是循环

重复做同样的事情对于人来说很累，但计算机不怕累，它最擅长的就是重复同样的步骤，我们可以把这些活都交给它做。我们通过**循环语句**（loop）让计算机重复执行某个任务。

Python 的循环语句包括 for 循环和 while 循环，for 循环固定循环次数，而 while 循环则根据特定的条件而决定是否继续循环。

这就好比我要一个机器人去操场跑步，第一种循环指令要求机器人跑 5 圈，跑完 5 圈任务就完成了，这是计数循环 for 循环。第二种指令要求机器人一直跑步，直到铃声响起为止，在这种情况下，我们并不知道机器人会去跑多少圈，有可能跑 1 圈，铃就响了，也有可能跑 20 圈，铃还没有响，这种根据条件而决定是否执行的循环是 while 循环。



7.3　For 循环

7.3.1　重复打印任务

编写 Python 代码，把“感觉自己棒棒哒”这个字符串一行接一行打印 5 次。

实例

```
感觉自己棒棒哒
感觉自己棒棒哒
感觉自己棒棒哒
感觉自己棒棒哒
感觉自己棒棒哒
```

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

感觉自己棒棒哒 1

```
print("感觉自己棒棒哒")
print("感觉自己棒棒哒")
print("感觉自己棒棒哒")
print("感觉自己棒棒哒")
print("感觉自己棒棒哒")
```

对初学者而言，它只需手动复制粘贴即可，看上去简单，但一旦要处理的数据增大，重复次数多了，缺点就显而易见了。如果要打印 200、2000 次，难道我要复制 200、2000 次 print("感觉自己棒棒哒")，岂不手酸？如果我刚好需要重复 2673 次，为了检查代码是否正确，我还得一行一行数，非常麻烦。

此时，我们这章的主角 for 循环可以上场了，循环能够帮助我们重复执行某些语句，我只要在 range()的括号里填入 5，电脑便自动执行下面的代码块 5 次。

感觉自己棒棒哒 2

```
for i in range(5):
→   print("感觉自己棒棒哒")
```

使用 for 循环和 range()函数，对比算法 1 代码的行数大大减少，并且只需要修改 range(5)括号里的数字，就可以灵活改变循环的次数。

我们来总结 for 循环的结构：

```
for 循环变量 in range(循环次数):
    做某些事
for i in range(5):
    print("感觉自己棒棒哒")
```

循环的优势在于可以帮我们节省编程的时间及代码存储的空间，也可以让代码更加简洁、易读。

这是第一个例子。在这个例子里，我们通过 for 循环和 range(5)告诉计算机要执行 5 次。我们来看下面一个略有不同的例子。

7.3.2 案例 1：敌军还有 5 秒到达战场

我想编写一个 5v5 战斗的游戏，在游戏开始之前，需要点个名，让 5 位英雄战友各自说出“××已做好战斗的准备”这句话。

假设团员的名称都存在 hero_list 列表里，请编写程序实现下面的效果。

hero_list = ["刘备", "王昭君", "诸葛亮", "李白", "貂蝉"]

实例

```
欢迎来到决斗战场，敌军还有 5 秒到达战场。
全军出击！
刘备已做好战斗的准备。
```

```
王昭君已做好战斗的准备。
诸葛亮已做好战斗的准备。
李白已做好战斗的准备。
貂蝉已做好战斗的准备。
```

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好，尝试一下吧。

敌军还有 5 秒到达战场 1

1. 用 print()打印欢迎玩家的句子
2. 用 print()打印全军出击!
3. 列表 hero_list 中有 5 位英雄的名字。
4. 打印第一个英雄已做好战斗的准备
5. 打印第二个英雄已做好战斗的准备
6. 打印第三个英雄已做好战斗的准备
7. 打印第四个英雄已做好战斗的准备
8. 打印第五个英雄已做好战斗的准备

```
print("欢迎来到决斗战场，敌军还有 5 秒到达战场。")
print("全军出击！")
hero_list = ["刘备", "王昭君", "诸葛亮", "李白", "貂蝉"]
print(hero_list[0] + "已做好战斗的准备。" )
print(hero_list[1] + "已做好战斗的准备。" )
print(hero_list[2] + "已做好战斗的准备。" )
print(hero_list[3] + "已做好战斗的准备。" )
print(hero_list[4] + "已做好战斗的准备。" )
```

编好后，测试完我们来思考一下，这五行代码有重复，是否有更好的方法，让列表里的每一项都自动重复一遍呢?

```
print(hero_list[0] + "已做好战斗的准备。" )
print(hero_list[1] + "已做好战斗的准备。" )
print(hero_list[2] + "已做好战斗的准备。" )
print(hero_list[3] + "已做好战斗的准备。" )
print(hero_list[4] + "已做好战斗的准备。" )
```

Python 的设计师也考虑到了这一点，不仅是列表，甚至是字符串里的字符或是字典里的每一项都可以直接逐个重复。我们来看看第二种解法，使用逐个重复的列表循环。

敌军还有 5 秒到达战场 2

1. 用 print()打印欢迎玩家的句子
2. 用 print()打印“全军出击!”
3. 列表 hero_list 中有 5 位英雄的名字。
4. 从列表 hero_list 中一个个读取每一位英雄
   1）打印英雄和“已经做好战斗准备”连接的字符串。

```
1. print("欢迎来到决斗战场，敌军还有 5 秒到达战场。")
2. print("全军出击！")
3. hero_list = ["刘备", "王昭君", "诸葛亮", "李白", "貂蝉"]
4. for hero in hero_list:
   → print(hero + "已做好战斗的准备。")
```

虽然只有一个 print()，但句子“某英雄已做好战斗的准备”却被打印了 5 次。

我们在这里使用了 for hero in hero_list 这个列表循环语句，列表循环会从列表的开头循环到列表末尾，hero_list 列表有多少项，for 循环便会重复多少次。当 hero_list 列表里有 5 个列表项，循环的语句就执行 5 次。for 循环同样要缩进，for 循环里面缩进的一块代码为“循环体”，每次循环都会执行它。

在这里，hero 是个循环变量，跟前面例子里的 i 类似。它按顺序依次指代 hero_list 里面的每一项。第一次循环时它指代列表的第一项“刘备”，第二次循环时它指代列表的第二项“王昭君”，以此类推。

循环次数	hero	循环代码	效果
循环第 1 次	“刘备”	print(hero + "已做好战斗的准备。")	“刘备已经做好战斗的准备。”
循环第 2 次	“王昭君”		“王昭君已经做好战斗的准备。”
循环第 3 次	“诸葛亮”		“诸葛亮已经做好战斗的准备。”
循环第 4 次	“李白”		“李白已经做好战斗的准备。”
循环第 5 次	“貂蝉”		“貂蝉已经做好战斗的准备。”

hero 只是一个代名词，我们可以给它命名为任何符合变量名标准的名字。尝试修改一下 hero 的名字，效果将一样。

```
for haha in hero_list:
    print(haha + "已做好战斗的准备。")
```

总结列表循环的结构，如下所示。

```
for 循环变量 in 循环列表:
    做某些事
```

```
for hero in hero_list:
    print(hero + "已做好战斗的准备。")
```

7.3.3　for 循环语法

不仅是列表，字符串、元组等数据都可以通过 for 循环逐个重复，这种逐个重复的方式称为“遍历”。我们来看看其他的数据类型是如何遍历的。

for 语句遍历元组	`t = (0, 1, 2, 3, 4)`
	`for i in t:`
	`    print(i)`
	`0`
	`1`
	`2`
	`3`
	`4`

for 语句遍历列表	`hero_list = ["鬼谷子", "哪吒", "诸葛亮", "大乔"]`
	`for hero in hero_list:`
	`    print(hero + "已做好战斗的准备")`
	鬼谷子已做好战斗的准备
	哪吒已做好战斗的准备
	诸葛亮已做好战斗的准备
	大乔已做好战斗的准备

for 语句遍历字符串	`s = "Python"`
	`for i in s:`
	`    print(i)`
	`P`
	`y`
	`t`
	`h`
	`o`
	`n`

for 语句固定次数循环	`for i in range(0,5,2):`
	`    print(i)`
	`0`
	`2`
	`4`

在了解了不同的循环之后，我们来做练习吧。

7.3.4　案例 2：乘法口诀表

利用 for 循环语句，编写代码，打印数字 3 的乘法口诀表，如下。

实例

3 × 0 = 0
3 × 1 = 3
3 × 2 = 6
3 × 3 = 9
3 × 4 = 12
3 × 5 = 15
3 × 6 = 18
3 × 7 = 21
3 × 8 = 24
3 × 9 = 27

分析：这个程序中，用了 print()，有包含数字 0～9 的列表 numbers，还使用了列表循环 for…in…语句。

请你先尝试一下吧。

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

乘法口诀表 1

1. 列表 numbers 中有 0～9 共 10 个数字
2. 从列表 numbers 中一个一个读取每一个数字
3. 打印 3 乘以每一个数字得到结果的算式

```
1. numbers = [0,1,2,3,4,5,6,7,8,9]
2. for i in numbers:
3. →  print('3 ×', i,'=', 3 * i)
```

点击【Run】→【Run Module】执行程序。

循环次数	i	循环体	效果
第 1 次	0	print('3 ×', i,'=', 3 * i)	“3 × 0 = 0”
第 2 次	1		“3 × 1 = 3”
第 3 次	2		“3 × 2 = 6”
第 4 次	3		“3 × 3 = 9”
第 5 次	4		“3 × 4 = 12”
第 6 次	5		“3 × 5 = 15”

续表

循环次数	i	循环体	效果
第 7 次	6		“3 × 6 = 18”
第 8 次	7		“3 × 7 = 21”
第 9 次	8		“3 × 8 = 24”
第 10 次	9		“3 × 9 = 27”

编程练习：尝试编写程序把 1～9 乘法表打印出来。

7.3.5　range()函数

在上面的例子中，我们创建了包含一串数字的列表。对于循环次数少的任务我们可以直接创建列表，但假如要循环 100 次，10000 次呢?

```
numbers = [0,1,2,3,4,5,6,7,8,9,10,11,12,13…]
```

当循环次数变多后，通过创建列表来实现循环就太慢了，并且还很麻烦。我们可以使用 Python 里面的 range()函数来实现更方便的循环。

打开 Python Shell，输入下方代码，观察 print()的结果，回答一共循环了几次，最后一个数是多少?

`>>> for i in range(5):` `    print(i)`	`>>> for i in range(10):` `    print(i)`
循环了 5 次，从 0 到 4 打印	循环了 10 次，从 0 到 9 打印

根据测试结果，for i in range(x):就是循环 x 次，从 0 开始循环，循环到 x 的前一个数停止。

range()函数的操作看上去更方便，让我们用 range 函数改进乘法口诀表的代码吧。

乘法口诀表 2

1. 循环 10 次，i 从 0 开始循环到 10 结束，不包括 10
2. 打印 3 乘以 i 得到结果的算式

```
1. for i in range(10):
2. →  print('3 x', i,'=', 3 * i)
```

点击【Run】→【Run Module】执行程序，看看是否可以打印成功。

range()函数还有多种用法，我们可以看看下面几个例子。

（1）函数 range(x, y)

打开 Python Shell，输入下方代码，观察 print()的结果。

```
>>> for i in range(1,5):
        print(i)
1
2
3
4
```

```
>>> for i in range(3,7):
        print(i)
3
4
5
6
```

根据测试结果，for i in range(x, y):让循环变量 i 从 x 开始循环，到 y 开始停止循环，也就是在 y 之前结束循环。

（2）函数 range(x)

打开 Python Shell，输入下方代码，观察 print()的结果。

```
>>> for i in range(4):
        print(i)
0
1
2
3
```

```
>>> for i in range(0,4):
        print(i)
0
1
2
3
```

根据测试结果，for i in range(x):让循环变量 i 从 0 开始循环，到 x 开始停止循环，也就是在 x 之前结束循环。

（3）函数 range(x, y, z)

打开 Python Shell，输入下方代码，观察 print()的结果。

```
>>> for i in range(1,20,5):
        print(i)
1
6
11
16
```

```
>>> for i in range(3,30,7):
        print(i)
3
10
17
24
```

根据测试结果，for i in range(x, y, z):让循环变量 i 从 x 开始循环，到 y 开始停止循环，也就是在 y 之前结束循环，每个数之间间隔 z。

下表中总结了几种 range()函数的用法，以供参考。

代码	描述
for i in range(5):	循环 5 次，变量 i 从 0 到 5（不包括 5）
for i in range(0):	循环 0 次
for i in range(0,5):	循环变量 i 从 0 到 5（不包括 5）
for i in range(1,5):	循环变量 i 从 1 到 5（不包括 5）
for i in range(-2,3):	循环变量 i 从-2 到 3（不包括 3）
for i in range(-2,3,2):	循环变量 i 从-2 到 3（不包括 3），每次循环后 i+2
for i in range(0,-3,-1):	循环变量 i 从 0 到-3（不包括-3），每次循环后 i-1
for i in range(1,-10,-3):	循环变量 i 从 1 到-10（不包括-10），每次循环后 i-3

在了解了不同的 for 循环之后，我们来做练习吧。

7.3.6　for 循环练习

打开 IDLE，新建 Python 文档。新建下面 4 个列表。

```
numbers1 = [1,4,6,8]
numbers2 = [5,10,14,29]
names = ["Jerry", "Tom", "Sandy"]
adjectives = ["happy", "excited", "comfortable"]
```

请为下面的练习编写代码。

循环练习 1

编写一个循环，将 numbers1 的每一项 + 1，打印出来。

```
2
5
7
9
```

循环练习 2

编写一个循环，将 numbers1 的每一项加上 numbers2 的对应项，存进一个字符串，打印出来。

```
'6142037'
```

循环练习 3

编写一个循环，将 names 的每一项和 adjectives 里的对应项拼成一句话，打印出来。

```
Jerry is happy.
```

```
Tom is excited.
Sandy is comfortable.
```

循环练习 4

编写一个循环，将 names 的每一项和 adjectives 里的每一项，各拼成一句话，打印出来。

```
Jerry is happy.
Jerry is excited.
Jerry is comfortable.
Tom is happy.
Tom is excited.
Tom is comfortable.
Sandy is happy.
Sandy is excited.
Sandy is comfortable.
```

循环练习 1

1. 依次读取 numbers1 列表里的每一项，它的值用 i 表示。
   1）打印每一项的值+1 后的结果。

```
1. for i in numbers1:
   →  print(i + 1)
```

循环次数	i	循环体	效果
第 1 次	1	print(i + 1)	打印 2
第 2 次	4		打印 5
第 3 次	6		打印 6
第 4 次	8		打印 8

循环练习 2

1. 设变量 s 为空字符串。
2. 循环的次数是用 len 得到列表 numbers1 的总项数。循环从 0 开始到总项数结束，不包括总项数。
   1）变量 s 字符串每次连接上一次的变量 s 和本次循环中的两列表对应数字相加的值。
3. 打印变量 s。

```
1. s = ""
2. for i in range(len(numbers1)):
   →  s += str(numbers1[i] + numbers2[i])

3. print(s)
```

循环次数	i	numbers1[i]	numbers2[i]	numbers1[i]+numbers2[i]	s
第 1 次	0	1	5	6	'6'
第 2 次	1	4	10	14	'614'
第 3 次	2	6	14	20	'61420'
第 4 次	3	8	29	37	'6142037'

循环练习 3

循环的次数是用 len 得到列表 names 的总项数。循环从 0 开始到总项数结束，不包括总项数。

每次循环打印连接列表 names 索引值为 i 的字符串，字符串"is"，列表 adjectives 索引值为 i 的字符串和句号的字符串。

```
for i in range(len(names)):
→   print(names[i] + " is " + adjectives[i] + ".")
```

循环次数	i	names[i]	adjectives[i]	循环体	效果
第 1 次	0	Jerry	happy	print(names[i]+" is " +adjectives[i]+" .")	Jerry is happy.
第 2 次	1	Tom	excited		Tom is excited.
第 3 次	2	Sandy	comfortable		Sandy is comfortable.

循环练习 4

依次读取 names 列表里的每一项，它的值用 i 表示。

1）在 i 的循环中，依次读取 adjectives 列表里的每一项，它的值用 j 表示。

2）每次循环打印变量 i、" is"、j 和句号的字符串

```
for i in names :
→   for j in adjectives:
→   →   print(i + " is " + j + ".")
```

循环次数	i	j	循环体	效果
第 1 次	Jerry	happy	print(i + " is " + j + ".")	Jerry is happy.
第 2 次	Jerry	excited		Jerry is excited.
第 3 次	Jerry	comfortable		Jerry is comfortable.
第 4 次	Tom	happy		Tom is happy.
第 5 次	Tom	excited		Tom is excited.
第 6 次	Tom	comfortable		Tom is comfortable.
第 7 次	Sandy	happy		Sandy is happy.
第 8 次	Sandy	excited		Sandy is excited.
第 9 次	Sandy	comfortable		Sandy is comfortable.

7.4 案例 3：奶昔机器人

我利用闲暇时间和几个工程师伙伴制作了一个奶昔机器人，我负责给奶昔机器人编写软件程序。

我们给奶昔机器人设计了按钮。按下按钮，机器人会搅拌奶昔 10 下。利用我学过的 for 循环，我决定写下下面的程序，具体搅拌的动作会交给机械工程师去处理，我在这里先用 print()来代替它的搅拌动作。

```
for i in range(10):
→   print("搅拌第" + str(i + 1) + "下")
```

循环次数	i	i+1	循环体	效果
第 1 次	0	1	print("搅拌" + str(i + 1) + "下")	“搅拌第 1 下”
第 2 次	1	2		“搅拌第 2 下”
第 3 次	2	3		“搅拌第 3 下”
第 4 次	3	4		“搅拌第 4 下”
第 5 次	4	5		“搅拌第 5 下”
第 6 次	5	6		“搅拌第 6 下”
第 7 次	6	7		“搅拌第 7 下”
第 8 次	7	8		“搅拌第 8 下”
第 9 次	8	9		“搅拌第 9 下”
第 10 次	9	10		“搅拌第 10 下”

我们将做好的奶昔机器人给顾客使用，顾客用了几天说，按一下按钮，才搅拌 10 下，奶昔不均匀，希望我们改进产品。

我仔细想了想，确实每次搅拌固定的次数的设计并不理想，有时搅拌 10 下次数太少，奶昔不够均匀；而有时固定搅拌 1000 下，如果奶昔在 200 下时已均匀，后面的 800 下既浪费电又浪费用户时间。

了解了第一代奶昔机器人的缺陷后，我们开发了第二代奶昔机器人，新增了一个白色的旋钮开关。将旋钮向右拧，奶昔机器人开始工作，直到用户觉得搅拌均匀了，将旋钮向左拧，奶昔机器人停止工作。

想好方案后，我需要把这个方案的程序编写出来。在这种方案下，我们并不清楚要重复搅拌多少次，而是根据一个特定的条件（开关是否打开）决定是否重复执行搅拌动作，这就要用到 while 循环。

我们用 switchOn=True 表示开关打开，switchOn=False 表示开关关闭。有一个 checkSwitch()函数，只要开关开着，它就会返回 True；如果用户把开关关了，那么它会返回 False。奶昔机器人每搅拌一次后会用 checkSwitch()检查开关状态，如果开关开着，会持续执行循环，否则会停止循环。

```
while switchOn == True:
→   print("搅拌一下")
→   switchOn = checkSwitch()
```

顾客们收到了第二代奶昔机器人后对我们大加赞扬，我们也很开心，了解了不同的循环。

```
for i in range(10):
→   print("搅拌第" + str(i + 1) + "下")

#奶昔机器人第一代：固定搅拌 10 下
```

```
while switchOn == True:
→   print("搅拌一下")
→   switchOn = checkSwitch()
#奶昔机器人第二代：一直搅拌，直到开关关闭为止
```

奶昔机器人给我们的启示是，如果不确定代码块需要重复执行多少次，我们可以使用 while 循环，只要满足某种条件就持续循环。下面我们来详细学习 while 循环。

7.5　while 循环

7.5.1　while 循环的意义

在日常生活中，我们经常根据条件，一直做某件事。

- 只要没到退休，就要一直工作。
- 只要学生没毕业，就要一直去上学。

```
while graduated == False:
→   students.goToSchool()
```

- 只要没满 18 岁，就不能考取驾照。

```
while age < 18:
→   print("你还不能考驾照哦，再等一年吧")
→   print("一年之后")
→   age += 1
```

希望这几个例子能够帮助你了解 while 循环的意义。我们再来比较 while 循环和 if 循环，看看它们的不同。

7.5.2 比较 while 和 if

观察下列语句，比较 while 和 if 的结果。

```
while 我很口渴：                    if 我很口渴：
    喝一口水                            喝一口水
#只要口很渴，就重复喝一口水          #如果口渴，就喝一口水
```

while 和 if 的语句结构类似。根据上面的例子，使用 while 时，只要“我很口渴”的条件满足，就会持续执行“喝一口水”这个动作，直到“我很口渴”的条件判断为 False 时结束。而使用 if 时，如果“我很口渴”的条件满足，则会喝一口水，喝完这口水后我是否还口渴就不重要了。

再看看下面的例子：

代码	age = 14 while age < 18: → print(str(age) + "岁不能考驾照哦，再等一年吧") → age += 1	age = 14 if age < 18: → print(str(age) + "岁不能考驾照哦")
测试结果	14 岁不能考驾照哦，再等一年吧 15 岁不能考驾照哦，再等一年吧 16 岁不能考驾照哦，再等一年吧 17 岁不能考驾照哦，再等一年吧	14 岁不能考驾照哦
描述	对于 while 来说，只要条件 age < 18 为 True，就持续执行循环体，直到 age < 18 为 False 为止	对于 if 来说，如果 age<18 判断为 True，则执行里面的代码块，否则什么都不做

循环次数	age	条件 age < 18	条件判断结果	效果
第 1 次	14	14 < 18	True	运行循环体 打印“14 岁不能考驾照哦，再等一年吧”，age + 1
第 2 次	15	15 < 18	True	运行循环体 打印“15 岁不能考驾照哦，再等一年吧”，age + 1
第 3 次	16	16 < 18	True	运行循环体 打印“16 岁不能考驾照哦，再等一年吧”，age + 1
第 4 次	17	17 < 18	True	运行循环体 打印“17 岁不能考驾照哦，再等一年吧”，age + 1
第 5 次	18	18 < 18	False	离开循环

7.5.3　while 循环语法

我们来总结 while 循环的结构：

```
while 判断条件：
    做某些事
```

当 while 后的判断条件为 True 时，就会进入循环体反复执行代码，直到判断条件为 False 才停止循环，跳出循环体。while 的循环体也需要缩进。

下面我们来练习使用 while 循环编写程序。

7.5.4　案例 4：加血道具的回血

在我开发的 5v5 游戏中，有一个加血道具叫云南灵药。玩家使用后会处于一个回血状态，每隔一秒，血量 + 10，直至满血 100。请你尝试编写这个回血机制。

一开始打印玩家龙傲天血量还剩 60，打印玩家龙傲天使用云南灵药，之后每 1 秒钟血量+10，将当前血量打印出来。当玩家血量为 100 以后，打印血量已满。

实例

```
玩家龙傲天血量还剩 60
玩家龙傲天使用云南灵药
玩家龙傲天血量 + 10，为 70
玩家龙傲天血量 + 10，为 80
玩家龙傲天血量 + 10，为 90
玩家龙傲天血量 + 10，为 100
玩家龙傲天血量已满
```

在这个程序里，我们可以使用一个变量代表玩家的血量，比如 hp。之后用 while 循环编写血量增加的程序。

另外，我们需要使用 time 模块，帮我们计算时间，计算好 1 秒的间隔。用 import time 导入时间模块后，使用 time.sleep()的指令能够让代码延迟，比方说 time.sleep(1)会延迟 1 秒，time.sleep(12)会延迟 12 秒。

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

加血道具的回血

1. 用 import 导入 time 库
2. 用变量 hp 保存血量，初始值为 60
3. 用 print()打印剩余血量
4. 用 print()打印使用加血道具
5. while 循环判断变量 hp 是否小于 100
   1）循环体代码：等待 1 秒
   2）循环体代码：变量 hp 值+10
   3）循环体代码：打印最新血量值
6. 打印血量加满

```
1. import time
2. hp = 60
3. print("玩家龙傲天被动感光波击中，血量还剩" + str(hp))
4. print("玩家龙傲天使用云南灵药")
5. while hp < 100:
    → time.sleep(1)
    → hp += 10
    → print("玩家龙傲天血量 + 10，为" + str(hp))

6. print("玩家龙傲天血量已满")
```

点击【Run】→【Run Module】执行程序。

循环次数	hp	条件 hp< 100	条件判断结果	效果
第 1 次	60	60<100	True	等 1 秒，hp+10，打印“玩家龙傲天血量+10，为 70”
第 2 次	70	70<100	True	等 1 秒，hp+10，打印“玩家龙傲天血量+10，为 80”
第 3 次	80	80<100	True	等 1 秒，hp+10，打印“玩家龙傲天血量+10，为 90”
第 4 次	90	90<100	True	等 1 秒，hp+10，打印“玩家龙傲天血量+10，为 100”
第 5 次	100	100<100	False	离开循环

只要 hp 小于 100，就持续等待 1 秒并让 hp 变量增加 10，打印当前血量。当 hp 小于 100，判定为 False 时，则离开循环，运行循环外的下一行代码。

参考上面的例子，我们来尝试一下下面的编程挑战：

在我设计的游戏里有一个毒箭机关。玩家被毒箭机关射中后，会立刻扣掉 125 点血量，然后进入中毒状态，每秒扣中毒前血量 10%的血量，直到血量低于 300，不再扣血。请编写一个 720 血量的战士雅典娜受到毒箭机关的代码，效果如下：

实例

```
战士雅典娜血量 720
战士雅典娜被毒箭机关击中，血量还剩 595
战士雅典娜陷入中毒状态
战士雅典娜血量减少 10%，为 523.0
战士雅典娜血量减少 10%，为 451.0
战士雅典娜血量减少 10%，为 379.0
战士雅典娜血量减少 10%，为 307.0
战士雅典娜血量减少 10%，为 235.0
战士雅典娜中毒状态消失，血量剩余 235.0
```

根据你的想象，运用 while 循环，把这个场景编写出来吧！参考答案见本书附赠资料。

7.5.5　无限循环和 break 语句

while 循环根据条件决定是否持续循环。要是条件永远满足，是不是循环就永远继续呢？

我们来练习下面的例子，如果需要手动停止程序，同时按下 Ctrl+C 组合键即可。

代码	`import time`
	`countdown = 3`
	`print("剩下", countdown, "秒")`
	`while True:`
	`→ time.sleep(1)`
	`→ countdown -= 1`
	`→ print("剩下", countdown, "秒")`
		`import time`
		`countdown = 3`
		`print("剩下", countdown, "秒")`
		`while True:`
		`→ time.sleep(1)`
		`→ countdown -= 1`
		`→ print("剩下", countdown, "秒")`
		`→ if countdown == 0 :`
		`→ → break`

测试结果	剩下 3 秒 剩下 2 秒 剩下 1 秒 剩下 0 秒 剩下 -1 秒 剩下 -2 秒 ……	剩下 3 秒 剩下 2 秒 剩下 1 秒 剩下 0 秒

续表

描述	while True 循环语句是指无限循环，如倒计时中一直不断地执行循环体。没有任何条件终止循环	在 while True 循环语句中，可以使用条件判断和 break 语句跳出无限循环。满足倒计时为 0 的条件后停止

在左边的例子中，while 循环的条件 True 一直判断为 True，它会一直不停地“跑”程序，占用很多系统资源，甚至导致死机，我们需要一个能够停止它的办法。在右边的例子中我们加入了一个 break 语句，就能在任何我们想要的位置跳出 while 循环。break 的意思是“打破”，在程序中可以理解为“打破”这个循环。

小贴士：无限循环与死循环

在编程中，一个靠自身控制无法终止的程序称为“死循环”。

根据目的区分，死循环是不符合目的的无限循环。死循环是你不想让它一直循环，但是它就一直循环，只能强制关闭，这就是战术错误。

无限循环中除了死循环还有有价值的无限循环。比如用户做计算题，只要输入的答案不对就让用户一直输入，这就是战略需要。

简而言之，无限循环是战略目的，死循环是战术失误。

无限循环的语法总结如下，它需要和 break 语句搭配使用。

```
while True:
        做某些事
        if 判断条件 1:
            break
```

为什么我要给自己创造一个无限循环，而不是把循环条件直接列在 while 旁边？因为有时循环里有多个地方需要跳出循环。我来举一个例子说明一下，如果我的游戏是个大的 while 循环，在游戏运行时，有多种情况需要结束游戏跳出循环，用 break 语句能够灵活处理这些不同的情况。

```
while True:
        游戏继续
        if 时间到:
                break
        if 我方阵亡:
                break
        if 敌方阵亡:
                break
```

```
    if 用户点击离开游戏按钮:
        break
...
```

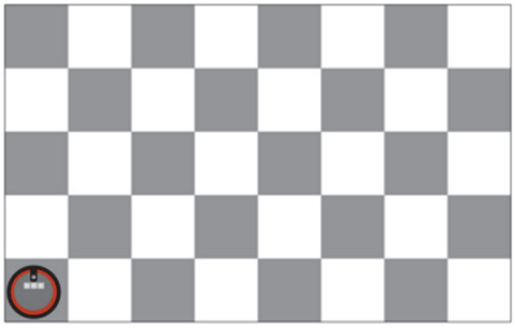
7.6　案例 5：扫地机器人的故事

好友知道我们学会编程后，会跟我们讨论有趣的想法。有一天，有一位同学说：“我们何不一起做个扫地机器人呢？”

我的机械工程师伙伴把扫地机器人的硬件做好了，为了简单快速地完成这个任务。这个机器人只能听懂下面两个基础指令，做这两件事。机器人每走到一个格子就会自动清扫这个格子。

指令	描述
move( )	前进一格
turn( )	右转 90 度

设计师设计了三个任务，如果扫地机器人都能成功完成，那么第一代机器人才算达标。作为负责编写程序的团队成员，我们尝试用 Python 给扫地机器人编写指令，完成以下三项任务。

		
任务一：要求机器人完成前方 4 个格子的清扫	任务二：要求扫地机器人完成图中 10 个格子的清扫	任务三：要求扫地机器人完成 8×5（长×宽）共 40 个格子的清扫

尝试在纸上用 move()和 turn()两个指令，以及我们学过的 Python 语法。写出让扫地机器人完成以上三个任务的程序吧。

扫地机器人的程序

任务一：要求机器人完成前方 4 个格子的清扫	循环 4 次前进一格	`for i in range(4):` `→ move()`

任务二：要求扫地机器人完成图中 10 个格子的清扫	1. 循环 4 次前进一格
	
	2. 右转 90 度
	3. 前进一格
	4. 右转 90 度
	5. 循环 4 次前进一格
		`for i in range(4):`
		`→ move()`
		`turn()`
		`move()`
		`turn()`
		`for i in range(4):`
		`→ move()`

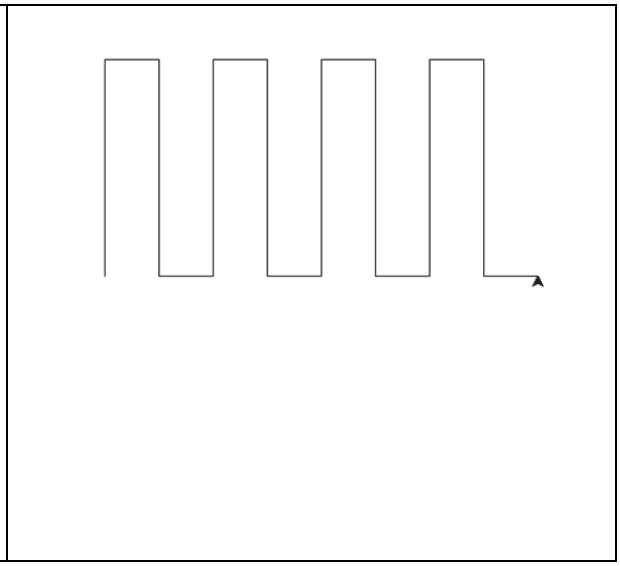
任务三：要求扫地机器人完成 8×5（长×宽）共 40 个格子的清扫	1. 下面 2～9 代码循环 4 次
	2. 循环 4 次前进一格
	
	3. 右转 90 度
	4. 前进一格
	5. 右转 90 度
	6. 循环 4 次前进一格
	
	7. 循环 3 次右转 90 度
	
	8. 前进一格
	9. 循环 3 次右转 90 度
		`for i in range(4):`
		`→ for i in range(4):`
		`→ → move()`
		`→ turn()`
		`→ move()`
		`→ turn()`
		`→ for i in range(4):`
		`→ → move()`
		`→ for i in range(3):`
		`→ → turn()`
		`→ move()`
		`→ for i in range(3):`
		`→ → turn()`

任务三：要求扫地机器人完成 8×5（长×宽）共 40 个格子的清扫	1. 下面 2～9 代码循环 2 次
	2. 右转 90 度
	3. 循环 7 次前进一格
	
	4. 循环 3 次右转 90 度
	
	5. 前进一格
	6. 循环 3 次右转 90 度
	
	7. 循环 7 次前进一格
	
	8. 右转 90 度
	9. 前进一格
	10. 右转 90 度
	11. 循环 7 次前进一格
		`for i in range(2):`
		`→ turn()`
		`→ for i in range(7):`
		`→ → move()`
		`→ for i in range(3):`
		`→ → turn()`
		`→ move()`
		`→ for i in range(3):`
		`→ → turn()`
		`→ for i in range(7):`
		`→ → move()`
		`→ turn()`
		`→ move()`
		`turn()`
		`for i in range(7):`
		`→ move()`

任务三我们给出了两种不同的走法，当然还有许多其他解法。今后在我们编写程序或者思考如何编写程序的过程中，不妨大胆尝试不同的解法，比较各个解法的优缺点。

我们可以用 Python turtle 验证算法，用 t.forward(50)代替 move()，用 t.right(90)代替 turn()，看看走出的路线是否符合要求。

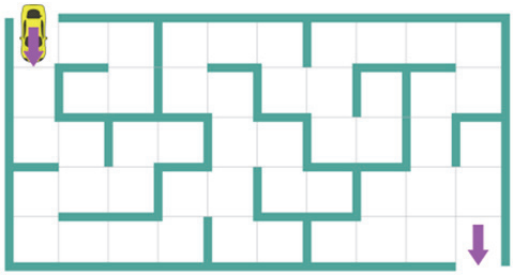
```
import turtle,time
t = turtle.Pen()
t.left(90)
for i in range(4):
    for i in range(4):
        t.forward(50)
    t.right(90)
    t.forward(50)
    t.right(90)
    for i in range(4):
        t.forward(50)
    for i in range(3):
        t.right(90)
    t.forward(50)
    for i in range(3):
        t.right(90)
```



7.7　案例 6：自动驾驶程序的故事

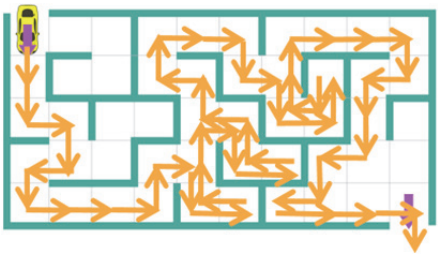
开发完扫地机器人后，我们又开发了一个自动驾驶程序。

请你根据右边的程序，画出兰博尼尼车的路线图，看兰博尼尼车是否可以走出迷宫？尝试根据右边的算法，画出跑车的路线吧。



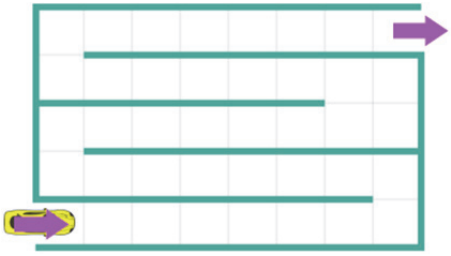
```
while 还没逃出迷宫:
    if 右边有路:
        turn()
        move()
    else:
        turn()
        turn()
        turn()
```

答案：



```
while 还没逃出迷宫:
    if 右边有路:
        turn()
        move()
    else:
        turn()
        turn()
        turn()
```

这里有一个迷宫，尝试用 move()和 turn()两个指令，以及我们学过的 Python 语法，带着跑车走出迷宫吧。



参考答案：

```
for i in range(2):
    for i in range(7):
        move()
    for i in range(3):
        turn()
    move()
    for i in range(3):
        turn()
    for i in range(7):
        move()
    turn()
    move()
    turn()
for i in range(7):
    move()
```

```
while 还没逃出迷宫:
    if 右边有路:
        turn()
        move()
    else:
        turn()
        turn()
        turn()
```

在这里，我们分别用 for 循环和 while 循环为兰博尼尼编写自动驾驶程序，走出迷宫。用 while 循环，不需要给出具体的循环数，一套算法可以套用不同的场景。事实上，这套算法称

为“右手法则”，是著名的解决迷宫问题的算法之一。

小贴士：摸墙算法

摸墙算法又称为“绕墙走算法”，是一种运用右手法则进行迷宫搜索的初级算法。简而言之，就是进入迷宫后，选择一个方向，用右手贴着墙壁一直走下去，如图 7-1 所示。

这个法则也有失灵的时候。因为它的前提是“入口和出口都在一条线段上”，也就是说这堵墙必须是连通的才行，而如果遇到“回字形”迷宫，则会出现绕了一圈返回原地的情况，如图 7-2 所示。

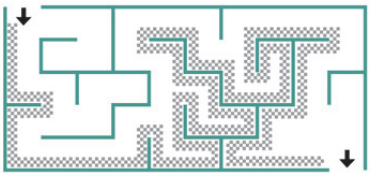


图 7-1　右手法则

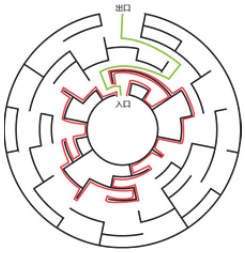


图 7-2　右手法则失败案例

7.8　案例 7：猜数字游戏

终于到了编写游戏的时刻！我们先从一个简单的猜数字游戏开始。

我们让计算机随机生成一个 0～100 的数字，让用户去猜。用户每次猜测后，程序将告诉用户猜的结果是大了、小了，或是猜对了。只要用户没猜对，游戏就一直继续，猜对后打印“游戏结束”。

这个小游戏并不长，却包含了字符串、数字、布尔值，以及条件判断和循环等编程概念，我们可以通过它回顾前面所学的概念。

这个游戏是这样运行的：

实例

```
请输入你的猜测（一个 0 到 100 之间的数字）>>>50
你猜得小了点
请输入你的猜测（一个 0 到 100 之间的数字）>>>75
```

```
你猜得大了点
请输入你的猜测（一个 0 到 100 之间的数字）>>>62
你猜得大了点
请输入你的猜测（一个 0 到 100 之间的数字）>>>56
你猜得大了点
请输入你的猜测（一个 0 到 100 之间的数字）>>>53
你猜得大了点
请输入你的猜测（一个 0 到 100 之间的数字）>>>52
你猜对了
游戏结束
```

在这个程序中，我们需要随机生成一个数，这会用到 random 随机模块。通过 import 导入 random 模块，用 random.randint(x, y)函数将返回 x 和 y 之间的任意整数，包括 x 和 y。

```
>>> import random
>>> random.randint(1,5)
5
>>> random.randint(1,5)
2
>>> random.randint(1000,2000)
1062
```

我们根据上面的运行结果梳理一下程序逻辑。

```
随机生成一个 0~100 之间的整数，存进变量 myNumber 里
用 while True 持续循环：
        让用户输入一个数字，存进变量 guess 里
        如果 guess 等于 myNumber：
                打印"你猜对了"
                用 break 离开循环
        否则如果 guess 大于 myNumber：
                打印"你猜得大了点"
        否则如果 guess 小于 myNumber：
                打印"你猜得小了点"
打印"游戏结束"
```

根据上面理清的程序逻辑，尝试自己编写代码吧。

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

猜数字游戏

1．用 import 导入 random 库
2．调用 random.randint()方法，得到一个 0~100 的整数，赋值给变量 myNumber
3．使用 while True 持续循环
　　1）询问用户猜测的数字，转成整数存入 guess 变量里
　　2）如果猜测数字等于随机生成的数字，打印“你猜对了”，跳出循环
　　3）否则如果猜测数字大于随机数字，打印“你猜得大了点”
　　4）否则如果猜测数字小于随机数字，打印“你猜得小了点”
4．打印游戏结束

```
1. import random
2. myNumber = random.randint(0,100)
3. while True:
→   guess = int(input("请输入你的猜测（一个 0 到 100 之间的数字）>>>"))
→   if guess == myNumber:
→   →   print("你猜对了")
→   →   break
→   elif guess > myNumber:
→   →   print("你猜得大了点")
→   elif guess < myNumber:
→   →   print("你猜得小了点")

4. print("游戏结束")
```

点击【Run】→【Run Module】执行程序，看看程序是否能正确运行。

小贴士：猜数字游戏技巧

在猜数字游戏中，最有效的方法是每次猜最中间的数，排除 1/2 的可能性。

这种算法叫二分查找法，又称折半查找法，如果有人要我猜一个 1~20 的数字，我会以下面这种折半查找的方式每次排除一半的可能性来得出答案，如图 7-3 所示。

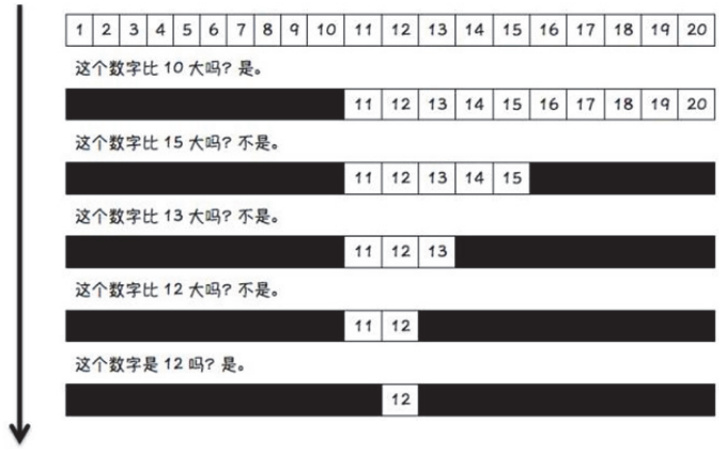


图 7-3　折半查找演绎

这种算法的优点是比较次数少，查找速度快，平均性能好。但需要数字按顺序排列好。

事实上，根据二分查找法，要猜一个 0 到 100 之间的数字，最多 7 次就可以猜出。我们可以尝试改进代码，添加一个变量 tryCount 记录玩家猜测的次数。

- 如果 tryCount<=7，打印，“你猜对了，共猜了 tryCount 次，棒棒的。”
- 如果 tryCount>7，打印，“你猜对了，共猜了 tryCount 次，还有更快猜出来的方法。”

运行效果如下：

```
请输入你的猜测（一个 0 到 100 之间的数字）>>>50
你猜得小了点
请输入你的猜测（一个 0 到 100 之间的数字）>>>75
你猜得大了点
请输入你的猜测（一个 0 到 100 之间的数字）>>>62
你猜得小了点
请输入你的猜测（一个 0 到 100 之间的数字）>>>68
你猜得小了点
请输入你的猜测（一个 0 到 100 之间的数字）>>>71
你猜对了，共猜了 5 次，棒棒的
游戏结束
```

自己尝试把这个挑战做出来吧，参考答案见本书附赠资料。

7.9　总结及课后练习

这一章我们了解了循环，学习了 for 循环和 while 循环。for 循环有固定的循环次数，而 while 循环则根据条件决定是否继续循环。

1. 编写一段代码，打印出 2 到 200 间的所有偶数，包括 2 和 200。

2. 编写一段代码，找出 0 到 1000 间所有 3 的倍数之和。

3. 将下面的 while 循环代码用 for 循环写出来。

```
a = 3
while a < 100:
→   print(a)
→   a += 2
```

第 8 章

抽象函数——分而治之的学问

本章重点是让读者了解函数，以及计算思维中的分解问题和抽象问题。读者会开始利用函数编写更加大型的程序，并了解递归函数的故事。

8.1 分而治之和抽象

在第 2 章我们学过用 turtle 模块画图形。若要画如图 8-1 所示的一朵花，会不会很难呢？

图 8-1 turtle 模块画出的花

你可以尝试一下。如果不记得 turtle 模块的语法，下面的代码可以参考。

```
import turtle  #导入海龟模块
t = turtle.Pen()  #给海龟命名为 t
t.forward(50) #让海龟 t 前进 50 步
t.left(90)  #让海龟 t 左转 90 度
t.right(6)  #让海龟 t 右转 6 度
```

如果觉得有难度，没有关系，接下来我们会教大家解决难题的方法。

画一整朵花看上去确实比较复杂，是一项巨大的任务。通过观察发现，我们可以将它拆分成 4 个稍微简单一些的小任务。

				
大任务	小任务 1	小任务 2	小任务 3	小任务 4

小任务 1 非常简单，直接向上画一条线就好。

```
import turtle
t = turtle.Pen()
t.left(90)
t.forward(50)
```

小任务 2 略微复杂，我们不知道叶子图形怎么画，怎么办？与其在这里忧心焦急，不如我们先跳过它，先把会做的部分完成。这里先假设这世界上有个 draw_leaf()指令，正好能帮我们画出叶子的形状。把 draw_leaf()放进来占位，之后再来解决这个问题。

```
import turtle
t = turtle.Pen()
t.left(90)
t.forward(50)
draw_leaf()
```

小任务 3 非常简单，又是向上画线，直接在 draw_leaf()后面把这行代码加进去即可。

```
import turtle
t = turtle.Pen()
t.left(90)
t.forward(50)
draw_leaf()
t.forward(150)
```

小任务 4 又有画叶子的内容了。不过仔细观察发现，其实就是把小任务 2 里的叶子重复画 6 遍，每次转 60°。既然 draw_leaf()是画一片叶子，那么用 for 语句循环 6 次就可以轻松画出来。

```
import turtle
t = turtle.Pen()
t.left(90)
t.forward(50)
draw_leaf()
t.forward(150)
for i in range(6):
    draw_leaf()
    t.right(60)
```

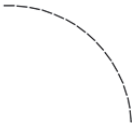
这一朵花的代码框架就这样完成了，感觉并不难！将画一朵花这个大任务分解成多个容易解决的小任务，这就是解题的一个重要技巧——**分而治之**。

等等！如果按以上代码运行，并不能画出花，因为世界上并没有 draw_leaf()这个指令呀！

我们把解题的大框架搭出来后，就可以忽略画花的任务，专心编写画叶子的这个指令了。同样可以用刚才提到的“分而治之”的方法。

一片叶子

将叶子分成：
画 2 个不同的弧形
每次画完后转 90°



将弧形分成：
画 15 条线
每次画完后转 6°

```
for i in range(2):
    for j in range(15):
        t.forward(5)
        t.right(6)
    t.right(90)
```

draw_leaf()是我们假设的一个指令，这个指令可以画叶子。在编程时，我们可以利用现成的指令，组合创造出新指令，这些指令称为函数（function）。我们可以用 def 关键词创造 draw_leaf()的指令：

```
def draw_leaf():
    for i in range(2):
        for j in range(15):
            t.forward(5)
            t.right(6)
        t.right(90)
```

把它添加进程序里就可以使用了！

右手一枝花

```
import turtle
t = turtle.Pen()

#定义绘制叶子的函数
def draw_leaf():
    for i in range(2):
        for j in range(15):
            t.forward(5)
            t.right(6)
        t.right(90)

t.left(90)
t.forward(50)
draw_leaf()  #调用绘制叶子的函数
t.forward(150)
for i in range(6):
    draw_leaf()  #调用绘制叶子的函数
    t.right(60)
```

我们把画叶子的程序隐藏在 draw_leaf()函数里。这种隐藏不必要细节的方法，就是**抽象**。draw_leaf()指令是我们定义的一个函数，可以隐藏不必要的细节，然后进行抽象。

下面我们来进一步学习函数的使用。

8.2　函数

通常，一个程序中有多个**函数**（function），每个函数完成某个小任务。其实 print()、range()都是之前用过的函数，它们有各自的功能。我们也可以通过把现成的指令捆绑在一起组成新的指令，定义新的函数。

有的函数需要我们输入某些信息，比如只输入 print()，计算机不知该打印什么，但是输入 print("早上好")计算机就看得懂，知道要打印“早上好”这个字符串。range()函数也需要参数，range(2,5)有两个参数，知道要从 2 开始数数，数到 5 时结束。这些输入的信息我们称之为**参数**。

函数通常都会有输出，有的输出是数值，比如 range(2,5)会给出一串数字；而有些输出是做动作，比如 print()会做出打印的动作。

函数让代码看起来更清晰，也方便我们重复使用代码，比如上一小节的 draw_leaf()函数就被调用了两次，每次使用时我只要把 draw_leaf()的名字写出来就好了，不需要每次完整写出两个 for 循环。如果我要修改代码，也只需修改函数里的代码即可。

我们来看看如何定义和调用函数。

8.2.1 定义并调用函数

在这里，我们定义了一个 print_greeting()的函数。之后我们只需写下 print_greeting()调用函数，它便会自动找到要做的两件事，即询问用户名和打印出带用户名的欢迎语句。

```
#定义函数
def print_greeting():
→   name = input("请输入您的用户名>>>")
→   print(name + ", 欢迎来到我的程序。")

#调用函数
print_greeting()
```

在 Python 中，我们使用关键词 def 定义函数，并将参数放在圆括号里，可以没有参数，或使用一个或多个参数。具体函数里的指令需要缩进，表示把代码块放进这个函数里。

```
def 函数名(参数列表):
    函数体
```

要使用时，则写出函数名和参数来调用函数。上面的例子里 print_greeting()没有参数，所以调用时括号里也是空的。

```
函数名(参数列表)
```

8.2.2 函数中代码的注意事项

前面我们介绍了函数的定义与调用，下面是函数中代码的注意事项。

（1）关键词 def 后面定义函数的部分会被主程序忽略，只有在调用函数时，程序才会跳到函数的那一行去执行函数内的代码。例如，下面的程序执行的顺序是 5、1、2、3。

```
1   def print_greeting():
2   →   name = input("请输入您的用户名>>>")
3   →   print(name + ", 欢迎来到我的程序。")
4
5   print_greeting()
```

（2）定义函数必须在调用函数之前。例如，先调用函数，再创建函数，程序会报错。

```
print_greeting()
def print_greeting():
    name = input("请输入您的用户名>>>")
    print(name + ", 欢迎来到我的程序。")

NameError: name 'print_greeting' is not defined
```

了解了基本的函数语法和运行规则后，我们来做一个小练习。根据 Python 函数的语法，如果运行了下面的代码，会打印出什么内容呢?

```
def sushi():
    print("把酒问青天")
def libai():
    print("床前明月光")
def wangwei():
    print("每逢佳节倍思亲")
def dufu():
    print("正是江南好风景")
def song():
    print("嘿嘿，嘿嘿，我来唱一唱")
song()
libai()
sushi()
song()
dufu()
wangwei()
wangwei()
```

根据各函数调用的顺序，打印出诗句。

```
嘿嘿，嘿嘿，我来唱一唱
床前明月光
把酒问青天
嘿嘿，嘿嘿，我来唱一唱
正是江南好风景
每逢佳节倍思亲
每逢佳节倍思亲
```

8.2.3　带参数的函数

之前，如果我们要打印某个数字的乘法口诀表，可以使用 for 循环完成任务。

实例

```
3 × 0 = 0
3 × 1 = 3
3 × 2 = 6
3 × 3 = 9
3 × 4 = 12
3 × 5 = 15
3 × 6 = 18
3 × 7 = 21
3 × 8 = 24
3 × 9 = 27
```

```
for i in range(10):
    print(3,"x", i , "=", 3 * i)
```

可惜这个代码只能写出 3 的乘法口诀表。要是想要做出 2 或者 4 乘法口诀表，我们则要替换代码里的所有 3。

改进前：需要替换 2 个 3	改进后：需要替换 1 个 3
for i in range(10): → print(3,"x", i , "=", 3 * i)	j = 3 for i in range(10): → print(j,"x", i , "=", j * i)

改进后，我们用变量 j 代替 3，这样只需改变变量 j 的数字，就可以替换乘法口诀表。

我们今天学习了函数，我预测乘法口诀会经常被用到，所以决定编写一个函数，专门打印某个数字的乘法口诀：

```
def multiply():
→  j = 3
→  for i in range(10):
→  →  print(j,"x", i , "=", j * i)

multiply() #调用函数
```

有了 multiply()函数，我们就可以直接反复使用它了。不过它有个致命的缺点！它现在只能打印 3 的乘法口诀。如果要打印其他数字的乘法口诀，则每次还得进去改函数。

这时候，参数的优点就体现了。我们来看看如何使用带参数的函数。我们把 j 提取出来变成参

数，这样每次调用 multiply 函数时，我们需要给它输入一个值，以这个值为参数 j，打印它的乘法口诀。

不带参数：函数打印 3 的乘法口诀	带参数：函数可以打印任何数的乘法口诀
def multiply(): → j = 3 → for i in range(10): → → print(j,"×", i , "=", j * i) multiply() #打印 3 的乘法口诀	def multiply(j): → for i in range(10): → → print(j,"×", i , "=", j * i) multiply(2) #j = 2，打印 2 的乘法口诀 multiply(3) #j = 3，打印 3 的乘法口诀 multiply(4) #j = 4，打印 4 的乘法口诀

有了这样灵活的函数，我们可以更加容易地打印出乘法口诀表。

```
def multiply(j):
→  for i in range(10):
→  →  print(j,"×", i , "=", j * i)

for num in range(1,10):
→  multiply(num)
```

我们来看看这个 for 循环，它的循环变量 num 将成为 multiply 函数的参数。我们调用了 multiply 函数 9 次，第一次调用 multiply(1)，到第 9 次调用 multiply(9)。

循环次数	num	multiply()的参数	调用函数	效果
第 1 次	1	1	multiply(1)	打印 1 的口诀表
第 2 次	2	2	multiply(2)	打印 2 的口诀表
第 3 次	3	3	multiply(3)	打印 3 的口诀表
第 4 次	4	4	multiply(4)	打印 4 的口诀表
第 5 次	5	5	multiply(5)	打印 5 的口诀表
第 6 次	6	6	multiply(6)	打印 6 的口诀表
第 7 次	7	7	multiply(7)	打印 7 的口诀表
第 8 次	8	8	multiply(8)	打印 8 的口诀表
第 9 次	9	9	multiply(9)	打印 9 的口诀表

有了参数，我们的函数将更加灵活。

我们再来看一个带参数的函数练习，编写一个函数 rmb_to_usd 将人民币转换成美元（假设

人民币兑换美元汇率为 6.798 : 1)。之后，编写程序并应用这个函数。首先询问用户想换多少人民币，然后调用函数打印出用户可以获得的美元数额。

实例

```
你想换多少人民币?>>>1000
你可以获得 147.10208884966167 美元
```

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口:【File】→【New File】），保存好。

人民币兑换成美元

1. 用 def 创建带 1 个参数 rmb 的 rmb_to_usd 函数
2. 用参数 rmb 计算出美元的数额，赋值给变量 usd
3. 用 print()打印可以获得的美元数额
4. 用 input()让用户输入人民币数额并存入变量 money 中
5. 调用 rmb_to_usd 函数，传入 1 个参数 money

```
1. def rmb_to_usd(rmb):
2. →   usd = rmb / 6.798
3. →   print("你可以获得", usd, "美元")
4. money = float(input("你想换多少人民币?>>>"))
5. rmb_to_usd(money)
```

点击【Run】→【Run Module】执行程序，确认函数和程序正确运行。

在这个例子里，我们使用了一个参数。事实上，我们创建函数是可以有多个参数的。我们将通过几个具体的练习熟悉函数的使用。

8.2.4 案例 1：简单的函数练习

函数练习 1：编写函数 cube(a)，打印出 a 的 3 次方。

```
>>>cube(2)
8
>>>cube(3)
27
>>>cube(-5)
-125
```

函数练习 2：编写函数 power(a, b)，打印出 a 的 b 次方。

```
>>>power(2,3)
8
>>>power(-3,4)
81
>>>power(-5,3)
-125
>>>power(1.01,100)
2.7048138294215294
#每天进步一点点，100 天后是之前的 2.7 倍。

>>>power(0.99,100)
0.36603234127322926
#每天懈怠一点点，100 天后剩下不到 4 成。
```

自己尝试把它做出来吧，记住上面 power(1.01,100)的原理！下面是参考答案和解释。

函数练习 1

1. 用 def 创建 cube 函数，带一个参数 a
2. 打印参数 a * a * a

```
1. def cube(a):
2. →   print(a * a * a)
```

函数练习 2

1. 用 def 创建 power 函数，带两个参数 a 和 b
2. 新建函数里的变量 num，赋值为 1
3. 使用 for 循环 b 次
4. 提取变量 num 值乘以 a 后再赋值给变量 num
5. 循环完成后打印 num 里现存的数字

```
1. def power(a,b):
2. →   num = 1
3. →   for i in range(b):
4. →   →   num *= a
5. →   print(num)
```

8.2.5　做事情的函数与返回值的函数

之前我们看到的函数都没有返回值。它们的输出是帮我们做某件事的。如下:

```
def draw_leaf():                    def rmb_to_usd(rmb):
```

```
for i in range(2):
    for j in range(15):
        t.forward(5)
        t.right(6)
    t.right(90)
#draw_leaf 函数的输出是画 30 条小直线
```

```
usd = rmb / 6.798
print("你可以获得", usd, "美元")
#rmb_to_usd 函数的输出是打印你可以获得的美元数额
```

比较下面两段代码，观察结果有什么区别?

```
def rmb_to_usd(rmb):
    usd = rmb / 6.798
    print("你可以获得", usd, "美元")
money = float(input("你想换多少人民币? >>>"))
rmb_to_usd(money)
#无返回值的函数，函数直接打印
```

```
def rmb_to_usd2(rmb):
    return rmb / 6.798

money = float(input("你想换多少人民币? >>>"))
rmb_to_usd2(money)
#没有任何内容打印出来
```

rmb_to_usd 函数直接将结果打印出来，我们无法继续使用算出的美元数额。而 rmb_to_usd2 函数并不会将结果打印出来。如果想要使用计算出的结果，需要将返回的值存入一个变量中，如下面代码中的 a，以便做进一步修改和使用。

```
def rmb_to_usd2(rmb):
    return rmb / 6.798
money = float(input("你想换多少人民币? >>>"))
a = rmb_to_usd2(money)
print(a)
```

```
你想换多少人民币?>>>1000
你可以获得 147.10208884966167 美元
```

函数返回给我们的数据叫做返回值（return value）。大多数情况下，我们传给函数一个参数，然后让它传给我们一个结果，在函数的末尾中使用关键词 return 返回数据。

```
return rmb / 6.798          #return 可以返回计算结果
return money                #return 可以返回变量
return 1.23                 #return 可以返回数字
return "Yes"                #return 可以返回字符串
return True                 #return 可以返回布尔值
```

使用返回值的优点在于让函数更加灵活，我们来看下面的例子。

我们的货币兑换功能受到了全球各国人民的欢迎，所以要设计四个不同语言的版本。下面是

两种不同的解法。

实例

货币兑换功能

方法 1：使用无返回值的函数。如果货币兑换功能需要支持 20 种语言，那就要写 20 个函数来显示不同的语言

```
def rmb_to_usd_ch(rmb):
→  usd = rmb / 6.798
→  print("你可以获得", usd, "美元")

def rmb_to_usd_en(rmb):
→  usd = rmb / 6.798
→  print("You will receive", usd, "dollars")

def rmb_to_usd_ja(rmb):
→  usd = rmb / 6.798
→  print("あなたは", usd, "ドルを獲得できます。")

def rmb_to_usd_ko(rmb):
→  usd = rmb / 6.798
→  print("너", usd, "달러를 얻을 수 있다。")

money = float(input("你想换多少人民币?>>>"))

rmb_to_usd_ch(money)
rmb_to_usd_en(money)
rmb_to_usd_ja(money)
rmb_to_usd_ko(money)
```

方法 2：使用有返回值的函数。如果货币兑换功能需支持 20 种语言，我们仍然只需要写 1 个函数来计算汇率，之后用不同的语言打印即可

```
def rmb_to_usd2(rmb):
→  return rmb / 6.798

money = float(input("你想换多少人民币?>>>"))

print("你可以获得",rmb_to_usd2(money), "美元")
print("You will receive ", rmb_to_usd2(money), "dollars")
```

```
print("あなたは",rmb_to_usd2(money), "ドルを獲得できます。")
print("너",rmb_to_usd2(money), "달러를 얻을 수 있다。")
```

如果不用返回值，同样的人民币兑换美元功能函数需要写四个，各函数里面用不同的语言打印出结果，十分冗赘。而如果使用返回值，这个函数将为我们返回出计算的结果，只需一个函数即可，之后我们想如何使用或打印这个数字都可以。就算是要编写 20 种语言，也只需要这一个函数。

8.3 案例 2：数学试卷机器人

数学老师每月要出几套数学试卷，十分辛苦。学了编程以后，我们希望能创建数学试卷机器人，帮助数学老师出题。让我们一起思考怎么编写这个机器人吧。

8.3.1 策划数学试卷机器人

我们先策划一个简单的版本，里面有三类数学题。首先，程序询问用户需要多少道题目。用户输入后，程序自动给出题目和标准答案。运行效果如下：

实例

```
请问这套试卷有多少道题目? >>>5
33 个人，每人吃了 5 个苹果，问总共吃了多少个苹果?
标准答案是 165 个
30 个人，每人吃了 4 个核桃，问总共吃了多少个核桃?
标准答案是 120 个
3 个人吃了 9 个蛋糕，问平均每人吃了多少个蛋糕?
标准答案是 3 个
14 个人，每人吃了 5 个面包，问总共吃了多少个面包?
标准答案是 70 个
长方体的边长为 65 厘米、96 厘米、74 厘米，它的体积是多少?
标准答案是 461760 立方厘米
```

我们从简单的版本开始，了解编写题库的程序逻辑，之后你可以发挥自己的想象，改进作品，让它成为更有用的题库程序。

因为程序需要随机选择及编写题目，所以在开始写代码之前，我们先了解随机模块的用法。

8.3.2　随机模块的用法

随机模块的函数可以给出随机数字。

random 模块里的函数	用途	例子示例
random.randint(x,y)	给出 x 和 y 间任意整数	>>>import random >>>print(random.randint(1,10)) 7
random.sample(range(x,y),n)	以列表方式给出 n 个 x 和 y 间任意整数，不包括 y	>>>import random >>>print(random.sample(range(1,10),2)) [5,7]
random.choice(myList)	给出 myList 列表里的任意项	>>>import random >>>myList = ["红色","绿色","蓝色"] >>>print(random.choice(myList)) 红色

在下一章里我们将深入学习 Python 的模块。我们先回到数学试卷机器人程序。

8.3.3　题目的函数

数字试卷机器人主要设置了三类题。你也可以设置更多类型的题目。我们来为每个类型的题编写一个函数。

第一类题

1	长方体的边长为 15 米、85 米、57 米，它的体积是多少?
2	长方体的边长为 88 米、60 米、81 米，它的体积是多少?
3	长方体的边长为 21 米、65 米、37 米，它的体积是多少?

上面 3 道题都是求体积，不同的是数据。编写函数 volume()，创建求体积的题目。其中，题目的 3 个数字是取 1 ~ 100 任意整数。

```
import random
def volume():
    a = random.randint(1,100)
    b = random.randint(1,100)
    c = random.randint(1,100)
    print("长方体的边长为",a,"米、",b,"米、",c,"米，它的体积是多少？")
```

```
volume() #调用 volume()函数，打印出一道题
volume() #再次调用 volume()函数，再打印出一道题
```

我们的代码多有重复，random.randint(1,100)重复了 3 次。事实上我们可以叠加给 a、b、c 同时赋值。注意，range 函数需要用 range(1,101)才可以给出从 1 到 100，包括 100 的任意整数。

```
import random
def volume():
    a, b, c = random.sample(range(1,101),3)
    print("长方体的边长为",a,"米、",b,"米、",c,"米，它的体积是多少？")

volume()
```

参考运行效果如下：

```
长方体的边长为 36 米、55 米、30 米，它的体积是多少？
```

我们希望每个题目附带显示答案。因此，可以添加一个变量 answer，算出体积并再编写一行代码，打印出答案。

```
import random
    def volume():
    a, b, c = random.sample(range(1,101),3)
    print("长方体的边长为",a,"米、",b,"米、",c,"米，它的体积是多少？")
    answer = a * b * c
    print("标准答案是",answer,"立方米")

volume()
```

参考运行效果如下：

```
长方体的边长为 95 米、32 米、96 米，它的体积是多少？
标准答案是 291840 立方米
```

为了给题目多些变化，我们可以任意选择单位，而不是每次都用“米”。在代码开头添加一个列表 units=["厘米","分米","米"]，然后修改函数，新建变量 unit，用 random.choice()让它赋值为 units 里的任意字符串，并且用它替换“米”。

```
import random
units = ["厘米","分米","米"]
def volume():
    a,b,c = random.sample(range(1,101),3)
    unit = random.choice(units)
```

```
→  print("长方体的边长为",a,unit,"、",b,unit,"、",c,unit,"，它的体积是多少？")
→  answer = a * b * c
→  print("标准答案是", answer, "立方", unit)


volume()
volume()
```

尝试运行代码，看是否能够打印出不同数字、不同单位的题目？

参考运行效果如下：

```
长方体的边长为 70 厘米、86 厘米、68 厘米，它的体积是多少？
标准答案是 409360 立方厘米
长方体的边长为 84 米、29 米、40 米，它的体积是多少？
标准答案是 97440 立方米
```

第二类题

有了第一类题目做参考，尝试编写函数 division()，自动生成类似下面的题目，随机选择人数、数量和食物，组成问题。

1	6 个人吃了 24 个梨子，问平均每人吃了多少个梨子？
2	31 个人吃了 62 个蛋糕，问平均每人吃了多少个蛋糕？
3	3 个人吃了 9 个苹果，问平均每人吃了多少个苹果？

这里有具体的提示和要求：

- 添加一个列表 foods = ["苹果","香蕉","蛋糕","核桃","梨子","面包"]
- 食物为 foods 中任意项
- 人数为 1～50 任意整数
- 每人食用数量为 1～5 任意整数

```
import random
foods = ["苹果","香蕉","蛋糕","核桃","梨子","面包"]
def division():
    #请你完成函数
division()
```

请尝试自己完成，division()函数的参考答案可在 8.3.5 小节的完整代码里找到。

第三类题

第三类题是乘法题，尝试编写代码，创建函数 times()，打印出“求所有人共吃多少食物”的题目。

1	9 个人，每人吃了 4 个面包，问总共吃了多少个面包?
2	2 个人，每人吃了 5 个梨子，问总共吃了多少个梨子?
3	11 个人，每人吃了 3 个苹果，问总共吃了多少个苹果?

这里有具体的提示和要求：

- 食物为 foods 中任意项
- 人数为 1～50 任意整数
- 平均每人食用数量为 1～5 任意整数

```
import random
foods = ["苹果","香蕉","蛋糕","核桃","梨子","面包"]
def times():
    #请你完成函数
times()
```

请尝试自己完成，times()函数的参考答案可在 8.3.5 小节的完整代码里找到。

8.3.4 策划程序逻辑

题目的函数已准备好，现在可以梳理程序的流程了。我们把流程分成以下三个步骤。

首先，用 input()让用户输入需要多少道题，将回答用 int()转换成整数，存进变量里。

```
num_of_questions = int(input("请问这套试卷有多少道题目? >>>"))
```

其次，用户需要多少道题，我们就要重复打印多少次，这里用到 for 循环，而次数就是 num_of_questions 里所存的题数。我们把 for 循环写出来。

```
num_of_questions = int(input("请问这套试卷有多少道题目? >>>"))
for i in range(num_of_questions):
    #打印任意题
```

最后，我们需要编写程序在 for 循环里打印任意题。我们先将打印题目的函数及它们所用到的数据添加进来。

新建 questions 变量，将 3 个函数存进一个列表里，将列表存进 questions 里。每次循环，我们将用 random.choice()任意选择一个函数，存进变量 question 里并调用这个函数。

```
import random
foods = ["苹果","香蕉","蛋糕","核桃","梨子","面包"]
```

```
units = ["厘米","分米","米"]
def volume():……
def times():……
def division():……
questions = [volume, times, division] #将 3 个函数存进 questions 列表里
num_of_questions = int(input("请问这套试卷有多少道题目? >>>"))
for i in range(num_of_questions):
→  question = random.choice(questions) #从 questions 选任意函数存入变量 question 里
→  question() #调用这个函数
```

三个步骤做好后，程序就完成了。下一小节里有完整的程序代码供大家参考。编写完程序后，不要忘了进一步思考如何改进这个题库的代码！

8.3.5　完整的程序代码

数学试卷机器人

```
import random
foods = ["苹果","香蕉","蛋糕","核桃","梨子","面包"]
units = ["厘米","分米","米"]
def volume():
→  a, b, c = random.sample(range(1,101),3)
→  unit = random.choice(units)
→  print("长方体的边长为", a, unit, "、", b, unit, "、", c, unit, "，它的体积是多少？")
→  answer = a * b * c
→  print("标准答案是", answer, "立方", unit)

def division():
→  persons = random.randint(1, 50)
→  items = random.randint(1, 5)
→  food = random.choice(foods)
→  total = persons * items
→  print(persons, "个人吃了", total, food, "，问平均每人吃了多少", food, "?")
→  print("标准答案是", items)

def times():
→  persons = random.randint(1,50)
→  items = random.randint(1,5)
→  food = random.choice(foods)
→  total = persons * items
→  print(persons, "个人，每人吃了", items, food, "，问总共吃了多少", food, "?")
```

```
    print("标准答案是", total)

questions = [volume, division, times] #将 3 个函数存进 questions 列表里

num_of_questions = int(input("请问这套试卷有多少道题目? >>>"))
for i in range(num_of_questions):
    question = random.choice(questions) #从 questions 选任意函数存入变量 question 里
    question() #调用这个函数
```

点击【Run】→【Run Module】执行程序。

编程挑战：编写口算测验机器人，询问用户要出几道题，题目为加法、减法、乘法，用户需要依次答题，最后统计正确率，计算出用户得分，以百分制计算。

```
试卷要几道题? >>>10
第 1 题:  5 + 7 = 12
第 2 题:  7 - 2 = 5
第 3 题:  9 - 5 = 4
第 4 题:  3 + 5 = 8
第 5 题:  7 * 9 = 63
第 6 题:  4 + 6 = 10
第 7 题:  3 - 7 = -4
第 8 题:  6 * 1 = 6
第 9 题:  6 - 3 = 3
第 10 题:  5 - 2 = 5
试卷全部答完，得分 90 分
```

8.4 递归函数的故事

8.4.1 阶乘与递归

递归函数是编程里特别有意思也比较有挑战的函数类型。在这一小节里，我们通过一个例子介绍递归函数，之后会给出几个递归函数相关的小项目，大家可以通过编程了解递归。

下面是一个编程挑战：

编写函数 roll(n)，n 是正整数，返回阶乘的结果 n * (n-1) * (n-2) * … * 2 * 1。例如 roll(5) 将返回 5 * 4 * 3 * 2 * 1，结果为 120。roll(8) 将返回 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1，结果为 40320。

大家通常会想到下面展示的方法 1，使用 for 循环重复往上乘。但今天我们来仔细研究方法 2，

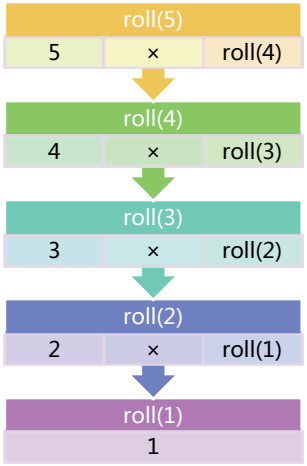
调用函数计算阶乘。

计算阶乘	
def roll(n): → current = 1 → total = 1 → for i in range(n): → → total = total * current → → current += 1 → return total print(roll(5)) 120	def roll(n): → if n == 1: → → return 1 → else: → → return n * roll(n - 1) print(roll(5)) 120
方法 1：使用 for 循环来计算阶乘	方法 2：调用函数计算阶乘

我们发现在方法 2 中，roll()函数在自己的定义里面被调用了。

在函数运行的过程中调用自己，就是递归。通常，我们用递归把规模大的问题转换为规模小的相似的子问题来解决。

在求阶乘的程序中，roll(5)=5*4*3*2*1 递归如下。



我们把大问题 roll(n)拆解成相似的小问题 roll(n-1)，当问题足够小的时候比如达到 roll(1)，roll(1)的答案就是 1，然后回到上一层，解决更大的问题。

8.4.2　无限递归

之前我们的结束条件是：如果到 1，就直接返回 1。如果没有明显的结束条件，就会一直递归下去。

```
def story():
→   return "从前有座山，山里有座庙，庙里头有个老和尚给小和尚讲故事，故事是:" + story()
print(story())
```

由于计算机的内存容量有限，无限递归很快会造成计算机死机，这时可以用 Ctrl+C 组合键退出。

8.4.3 案例 3：科赫曲线

科赫曲线（koch curve）运用递归原理画出的分形（fractal）图形。我们来看一下科赫曲线。

科赫曲线的构造方法：给定一个线段，把它等分三段，以三段的中间一段为底对齐，加入一个等边三角形，再去除该条线段。然后，对每个新线段重复进行上述步骤，就能形成科赫曲线。

第 0 级的科赫曲线是简单的线段。

koch(0)	

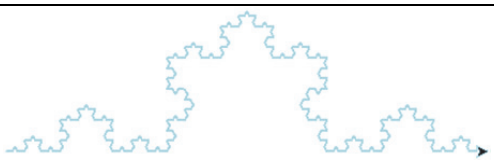
第 1 级的科赫曲线在线段中间凸起三角。

koch(1)	

第 2 级的科赫曲线在第 1 级的每条线段中间凸起三角。

koch(2)	

第 3 级的科赫曲线又在第 2 级的每条线段中间凸起三角，第 4 级等以此类推。

koch(3)	
koch(4)	

下面是科赫曲线递归函数代码，我们可以尝试编写以下代码。

科赫曲线

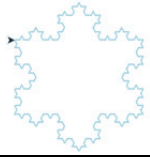
```
import turtle
t = turtle.Pen()
t.pencolor("lightblue")
t.width(2)
t.speed(1000)

def koch(order):
    if order == 0:
        t.forward(5)
    else:
        koch(order-1)
        t.left(60)
        koch(order-1)
        t.right(120)
        koch(order-1)
        t.left(60)
        koch(order-1)

koch(4)
```

8.4.4　案例 4：科赫雪花

如果画 3 条科赫曲线，每次旋转 120°，就能组成科赫雪花。下面给出科赫雪花的代码，大家可以尝试画出来。

koch(1)	koch(2)	koch(3)
		

科赫雪花

```
import turtle
t = turtle.Pen()
t.pencolor("lightblue")
t.width(2)
t.speed(1000)
```

```
def koch(order):
    if order == 0:
        t.forward(5)
    else:
        koch(order-1)
        t.left(60)
        koch(order-1)
        t.right(120)
        koch(order-1)
        t.left(60)
        koch(order-1)

for i in range(3): #重复执行3次画科赫曲线的代码可以得出科赫雪花
    koch(4)
    t.right(120)
```

8.5 变量的作用域

在这一章里，有些变量是在函数里定义的，而有些变量则是在大程序里定义的。变量按作用域划分为两种，“全局变量”和“局部变量”。定义在函数外的变量是全局变量，作用域是整个 Python 文件，所以称为“全局变量”。定义在函数内的变量是“局部变量”，作用域是在函数内部，所以称为“局部变量”。

在下面的例子中，usd 在函数 rmb_to_usd 里定义，是局部变量；而 money 在函数外定义，是全局变量。

```
def rmb_to_usd(rmb):
    usd = rmb / 6.798
    print("你可以获得", usd, "美元")
money = float(input("你想换多少人民币？>>>"))
rmb_to_usd(money)
```

如果全局变量和局部变量的名字相同会怎么样呢？尝试编写以下代码，观察结果。

房子里有多少钱

1. 设全局变量 money 值为 200
2. 定义 house()函数
3. 设函数内局部变量 money 值为 1000
4. 打印函数内的局部变量 money
5. 调用 house

6. 打印全局变量 money

```
1. money = 200
2. def house():
3. →  money = 1000
4. →  print("house()内:魔镜啊魔镜，房子里有多少钱? >>>",money)
5. house()
6. print("house()外:魔镜啊魔镜，房子外有多少钱? >>>",money)

house()内:魔镜啊魔镜，房子里有多少钱? >>> 1000
house()外:魔镜啊魔镜，房子外有多少钱? >>> 200
```

如果全局变量和局部变量的名字相同，那么在函数内则会使用局部变量。有关键词 global 的变量名才代表全局变量。

房子里有多少钱 2

1. 全局变量 money 值为 200
2. 定义 house()函数
3. 声名 money 变量为全局变量
4. 设置全局变量 money 值为 1000
5. 打印函数内的全局变量 money
6. 调用 house
7. 打印全局变量 money

```
1. money = 200
2. def house():
3. →  global money
4. →  money = 1000
5. →  print("house()内:魔镜啊魔镜，房子里有多少钱? >>>",money)
6. house()
7. print("house()外:魔镜啊魔镜，房子外有多少钱? >>>",money)

house()内:魔镜啊魔镜，房子里有多少钱? >>> 1000
house()外:魔镜啊魔镜，房子外有多少钱? >>> 1000
```

8.6　总结及课后练习

在这一章我们了解了函数，包括通过参数给函数输入信息，以及使用返回值输出信息。通过函数，我们可以将庞大复杂的程序分解成不同的功能块，帮助我们编写和阅读程序。我们还了解了在运行的过程中如何调用自己的递归函数。

1. 编写有返回值的函数 sum(a, b, c)，返回 a、b、c 三位数之和。

2. 编写函数 power(a)，返回 a 的 12 次方。

3. 编写水仙花数的函数 flower()，打印 100～1000 内所有的水仙花数。水仙花数是一个三位数，其各位数字立方之和等于该数字本身。注：^符号表示幂，5^3 表示 5 的 3 次方。

```
1 ^ 3+ 5 ^ 3+ 3 ^ 3 = 153
3 ^ 3+ 7 ^ 3+ 0 ^ 3 = 370
3 ^ 3+ 7 ^ 3+ 1 ^ 3 = 371
4 ^ 3+ 0 ^ 3+ 7 ^ 3 = 407
```

4. 编写程序来模拟概率实验。掷骰子 10000 次，统计得到各点数的概率。

```
骰子为 1 点的概率是 16.68 %
骰子为 2 点的概率是 16.66 %
骰子为 3 点的概率是 16.79 %
骰子为 4 点的概率是 16.38 %
骰子为 5 点的概率是 16.86 %
骰子为 6 点的概率是 16.63 %
```

第 9 章

Python 库——让强大的 Python 库帮忙

Python 的一个优势在于它强大的模块。本章重点是让读者了解 Python 模块的使用，并尝试用几个简单的库制作中型编程项目。

9.1 Python 模块概述

Python 的一大优势就在于它有许多强大的模块。这些模块如同巨大的函数资料库，存储着许多现成的函数让我们调用。这些函数会分门别类地放在不同的 Python 模块文件中，各有自己的功能。

比如我们之前用过随机模块，它其实是 random.py 文件，里面有许多函数，包括给我们返回随机整数的 randint(x,y)函数。使用 import 导入 random 模块，然后再通过 random.randint(1,10)得到 1 到 10 中的随机整数。

```
import random
print(random.randint(1,10)) #将会打印整数 1～10 中任意一个数，包括 1 与 10
```

random.randint(1,10)中的“点”可以理解为“的”。这句话的意思为调用 random 模块中的 randint(1,10)函数。

除随机模块以外，我们已经尝试过几个不同的模块。我们使用过 turtle 模块画图形和线条，使

用过 time 模块来延迟时间。

9.2 安装、卸载和使用 Python 模块

9.2.1 安装与卸载 Python 模块

Python 会自带许多模块，但有时我们也需要导入第三方模块。下面我们以 numpy 和 pygame 模块为例，教大家如何安装 Python 模块。建议大家先跳过这一小节，之后如果需要安装模块，再返回来看这一部分。

1. Windows 系统

（1）在 Windows 系统下，搜索 cmd，选择命令提示符，如图 9-1 所示。

（2）打开命令提示符窗口后，直接在箭头后面输入“python”这 6 个英文字母。如果计算机能成功识别“python”，则会显示 Python 版本，例如图 9-2 中的版本是 Python 3.6.3。

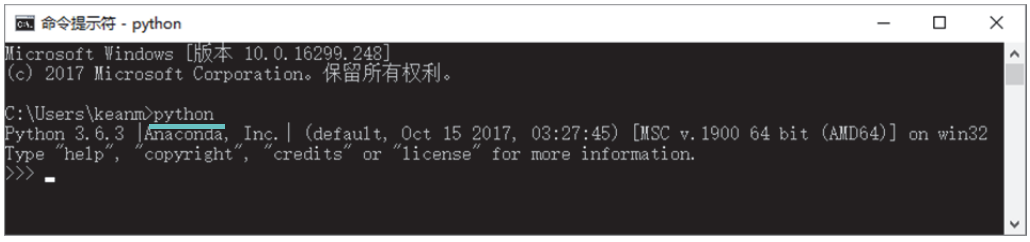


图 9-2 Windows 的 Python 命令行

注：如果不能识别 Python，则会显示“python”不是内部或外部命令。那么需要回到第 1 章，按照第 1 章的步骤安装 Python。

（3）能够识别 Python 后，最后一行将以>>>开头，此时你可以尝试在>>>后输入 Python 代码，效果相当于 Python Shell。

（4）接下来输入 exit()，退出 Python 环境，回到初始的路径。

（5）在路径地址结尾>后输入 pip install numpy 来让计算机自动下载安装 numpy 模块，如图 9-3 所示。pip install【模块名】指令将下载该模块。在这个环节要保证连网状态。如果该模块已经存在，则会显示请求已满足，否则将会弹出提示信息和下载进度条。

```
C:\Users\keanm>pip install numpy
Requirement already satisfied: numpy in d:\anaconda3\lib\site-packages
```

图 9-3　pip install 安装模块的指令

（6）安装好后，按照步骤 2 进入 Python，尝试 import numpy，如图 9-4 所示。如果没有任何新的信息出现，说明 numpy 模块已成功安装。

```
>>> import numpy
>>>
```

图 9-4　可以使用 numpy

（7）以后如果要卸载程序，则可以在路径结尾处输入 pip uninstall numpy 即可，如图 9-5 所示。

```
C:\Users\keanm>pip uninstall numpy
```

图 9-5　卸载 numpy

2. Mac OS 系统

（1）在 Mac OS 系统下，从 Finder 里选择【应用程序】→【实用工具】→【终端】，如图 9-6 所示，也可以通过 Spotlight 搜索找到终端。

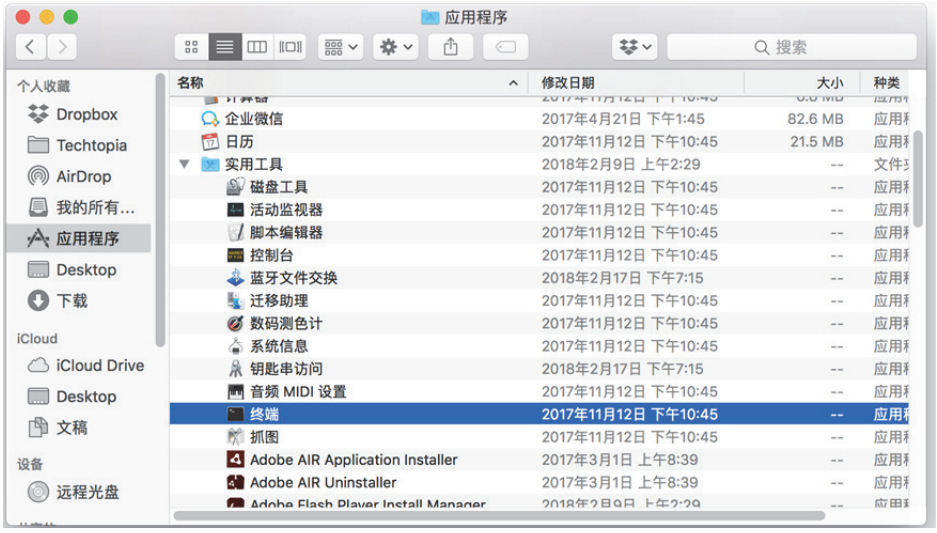


图 9-6　在 Mac 系统里找到“终端”

（2）打开终端窗口后，在$符号后面输入“python3”这 7 个字符。如果计算机能成功识别 Python，则会显示 Python 版本，图 9-7 中的版本是 Python 3.6.4。

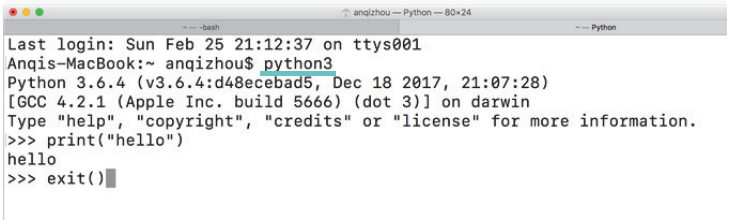


图 9-7　Mac 的 Python 命令行

（3）能够识别 Python 后，将进入 Python 命令行，你可以尝试在>>>后输入 Python 代码，效果相当于 Python Shell。

（4）输入 exit()退出 Python 环境，回到初始的路径。

（5）在路径地址结尾>后输入 pip3 install【模块名】将自动下载模块。比如 pip3 install pygame 会让计算机自动下载、安装 pygame 模块，如图 9-8 所示。在这个环节中要保证连网状态。如果该模块已经存在，则会显示请求已满足，否则将会弹出提示信息和下载进度条。

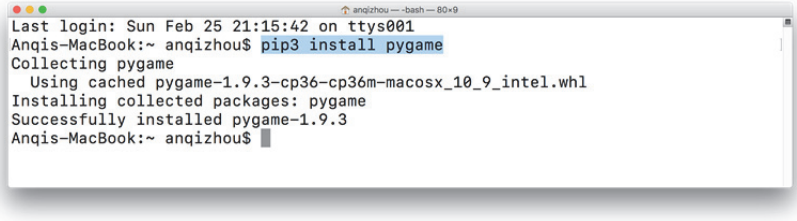


图 9-8　安装 pygame 模块的指令

（6）安装好后，按照步骤 2 进入 Python，尝试 import pygame，如果没有任何新的信息出现，如图 9-9 所示，则说明 pygame 模块已成功安装。

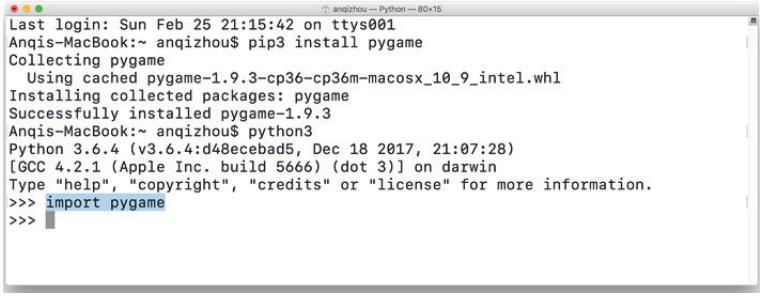


图 9-9　可以使用 pygame

（7）以后如要卸载程序，在路径结尾处输入 pip3 uninstall pygame 即可，如图 9-10 所示。

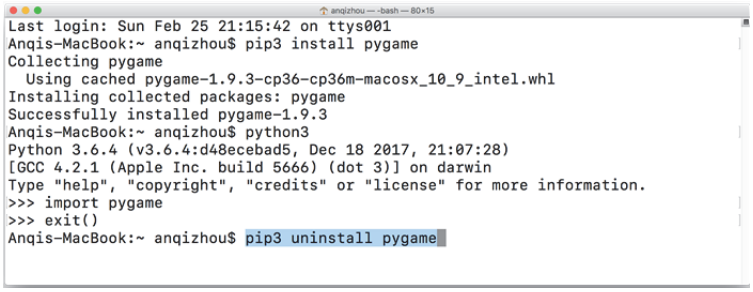


图 9-10　卸载 pygame 模块

了解了如何安装和卸载模块后，我们可以开始好好利用 Python 中无比强大的模块资料库了。在这一章里，我们将简单介绍几个模块的用法，但这些仅是 Python 库的九牛一毛。我们在下一小节里会介绍如何查找 Python 文档，希望大家之后能够根据自己的需求，灵活使用 Python 模块。

9.2.2　Python 文档

Python 文档是对 Python 语言的说明，比如，我们可以用来查询 Python 库中的函数，有哪些、怎么用，等等，可以把它当成 Python 的说明书。在这里我们给出两个常用文档的链接地址，在学习过程中如果遇到问题，这两个文档将成为你可靠的帮手。

Python 3.6.4 中文文档：http://www.pythondoc.com/pythontutorial3/index.html

Python 官方文档：https://docs.python.org/3/

Python 官方文档为英文版，我在这里简单举例介绍一下查询 random 模块函数的使用方法。

（1）点击官方文档右上角的 modules 选项可以进入相关页面，如图 9-11 所示。

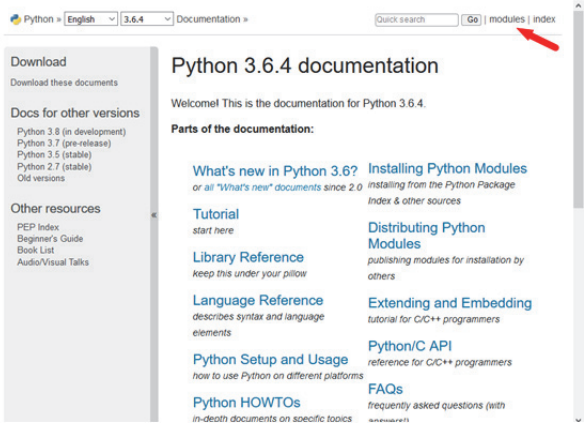


图 9-11　查找 Python 模块的文档

（2）页面里的目录将各模块按字母表顺序排列，可在 r 的部分找到 random 模块的使用方法，如图 9-12 所示。

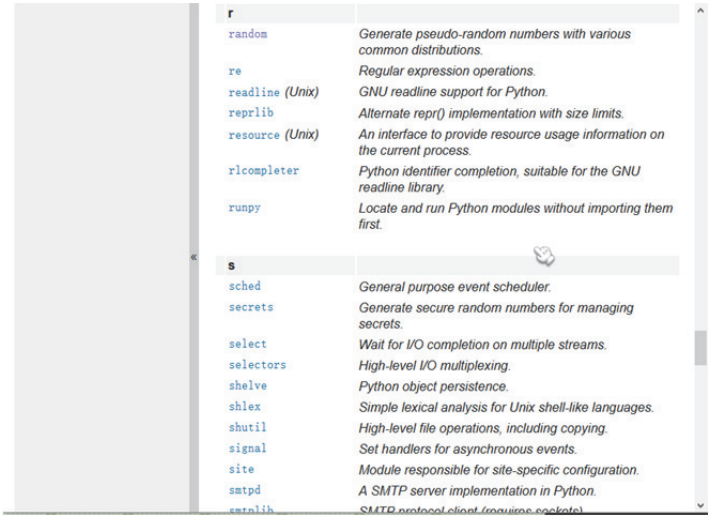


图 9-12　按字母表顺序排列的 Python 模块文档

（3）我们可以仔细阅读模块的介绍，也可以点击左边目录里的“9.6.7 Examples and Recipes”选项直接跳转到例子，如图 9-13 所示。

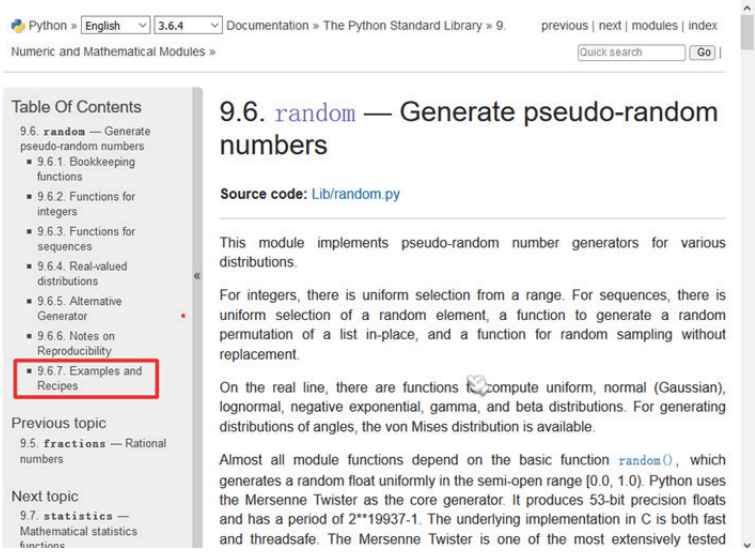


图 9-13　在 Python 模块文档中找寻例子

（4）根据举例，我们可以知道 random 中的函数有哪些，以及函数的用法，如图 9-14 所示。

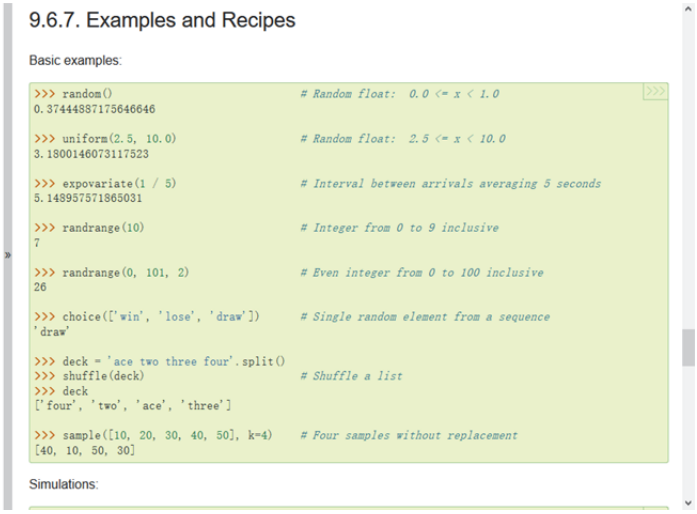


图 9-14　Python random 模块文档中的例子

了解了查询资料的方法，再加上大胆的尝试，相信你一定能灵活运用 Python 和它强大的模块了。下面我们将介绍几个有趣又常用的模块。

9.3　random 模块

9.3.1　随机模块常见函数

在第 8 章中，我们学习了随机模块的函数。下表列出了随机模块中的常见函数，输入 import random 后即可使用。

函数	作用	举例
random.random()	给出 0 至 1 之间的任意小数，不包括 1	random.random() 给出 0至1 之间的任意小数 如 0.5811562043603636
random.randint(a, b)	给出 a 至 b 之间的任意整数，包括 a 和 b	random.randint(1,10) 给出 1至10 间任意整数，包括 1 和 10 如 2
random.sample(range(a, b), n)	给出 n 个 a 至 b-1 之间的任意整数	random.sample(range(2, 200), 5) 给出一个列表，列表里有 5 项，都是 2至199 间任意整数 如[10, 29, 145, 34, 167]
random.choice(myList)	给出 myList 列表中的任意项	fruits = ["apple","pear","pineapple"] random.choice(fruits)

续表

函数	作用	举例
		给出 fruits 里面的任意一项 如"pineapple"
random.randrange(a, b, n)	给出 a 至 b-1 间每 n 个间隔的随机数	random.randrange(1, 20, 2) 给出 1 至 19 间每 2 个间隔的随机数 如 3
random.uniform(a, b)	给出 a 为下限 b 为上限的任意小数	random.uniform(1, 2) 给出 1 为下限 2 为上限的任意小数 如 1.2336851989113074
random.shuffle(myList)	将集合的所有元素打乱	numList = [1, 2, 3, 4, 5] random.shuffle(numList) 将列表 numList 的所有元素打乱 如[3, 2, 5, 4, 1]
random.sample(myList, n)	给出集合中任意 n 个元素	numList = [1, 2, 3, 4, 5] random.sample(numList, 2) 给出列表 numList 中任意 2 个元素 如[3, 2]

根据上面的常见函数，完成下面几个编程练习。

9.3.2 随机模块函数练习

1. 编写代码，打印出 100～200 的任意整数。

2. 编写代码，打印出 3 个 50～100 的任意整数。

3. 编写代码，打出 10～11 的任意小数。

4. 编写代码，打出 100～200 的任意偶数（双数）。

随机模块练习 1

1. 用 import 导入 random 模块
2. 随机选 1 个 100 至 200 之间的整数，赋值给 a
3. 打印 a

```
1. import random
2. a = random.randint(100,200)
3. print(a)
```

随机模块练习 2

1. 用 import 导入 random 模块
2. 随机选 3 个 50 至 100（101 前一位）之间的整数集合赋值给 b
3. 打印 b

```
1. import random
2. b = random.sample(range(50,101),3)
3. print(b)
```

随机模块练习 3

1. 用 import 导入 random 模块
2. 随机选 0 到 1 之间的小数赋值给 c
3. 提取 c 加上 10 再赋值给 c
4. 打印 c

```
1. import random
2. c = random.random()
3. c += 10
4. print(c)
```

随机模块练习 4

1. 用 import 导入 random 模块
2. 从 100 到 201 间每隔一个数的整数任选一个数字赋值给 d
3. 打印 d

```
1. import random
2. d = random.randrange(100,201,2)
3. print(d)
```

9.3.3　案例 1：幸运大抽奖

模特小熊霸霸在微博上发起了一个转发博文抽奖得手机的活动，最终只抽取 2 名幸运粉丝获得手机。所有转发博文的粉丝皆可参与，粉丝的名字存在变量 a 里。

编写一个幸运大抽奖程序。从粉丝列表中抽取 2 名幸运儿。编写函数 luckydraw，带参数 myList，当执行 luckydraw(a)时，会返回 a 里的一位随机人物。

实例

["吃瓜路人", "菇凉凉", "随遇安", "Sunny", "未央", "武林萌主"]
随遇安和未央获奖。

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

幸运大抽奖

1. 用 import 导入 random
2. 列表 a 中有 6 个字符串，各为人物名
3. 初始化变量 name1 与 name2 为空字符串
4. 定义函数 luckydraw，带一个参数 myList
   1）返回 myList 中任意一项
5. 判断 name1 与 name2 的值，若相等则进入循环
   1）调用 luckydraw(a)赋值给 name1
   2）调用 luckydraw(a)赋值给 name2
6. 打印 a 列表
7. 打印 name1，name2 获奖

```
1. import random
2. a = ["吃瓜路人", "菇凉凉", "随遇安", "Sunny","未央","武林萌主"]
3. name1 = name2 = ""
4. def luckydraw(myList):
   →  return random.choice(myList)

5. while name1 == name2:
   →  name1 = luckydraw(a)
   →  name2 = luckydraw(a)

6. print(a)
7. print(name1 + "和" + name2 + "获奖。")
```

点击【Run】→【Run Module】执行程序。

这并不是唯一的解法，我们还可以换一种方式，用 random.sample 函数解题。

```
import random
a = ["吃瓜路人", "菇凉凉", "随遇安", "Sunny","未央","武林萌主"]
name1, name2 = random.sample(a, 2)
print(a)
print(name1 + "和" + name2 + "获奖。")
```

完成后，请尝试一下下面的挑战：

粉丝将博文转发后，每个人获得的点赞数不同，为了鼓励更多人点赞，我们需要改进抽奖机制。点赞数目越多的人被抽到的可能性越大。请编写幸运大抽奖第二代，计算机询问用户中有谁抽奖，点赞数多少。抽取 2 名幸运粉丝，能够实现点赞数越多的人抽中奖品的可能性越大。

实例

```
请输入已转发用户?(输入-1 结束) >>>武陵人捕鱼
请输入武陵人捕鱼的点赞数? >>>5
请输入已转发用户?(输入-1 结束) >>>吃瓜小姐姐
请输入吃瓜小姐姐的点赞数? >>>15
请输入已转发用户?(输入-1 结束) >>>蓝瘦香菇
请输入蓝瘦香菇的点赞数? >>>8
请输入已转发用户?(输入-1 结束) >>>-1
吃瓜小姐姐和武陵人捕鱼获奖
```

自己尝试做出来吧，参考答案见本书附赠资料。

9.4　时间模块和日期时间模块

9.4.1　时间模块

Python 时间（time）模块提供用来处理时间的函数。时间模块用 import time 导入，下面是几个常见的函数：

函数	作用	举例
time.time()	返回这一刻的时间戳 stamp	>>>time.time() 1519652795.2386184
time.localtime(stamp)	把时间戳转换为当前时区的时间元组	>>>time.localtime(1519652795.2386184) time.struct_time(tm_year=2018,tm_mon=2,tm_mday=26, tm_hour=21,tm_min=46,tm_sec=35,tm_wday=0, tm_yday=57,tm_isdst=0)
time.ctime(stamp)	把时间戳 stamp 转换为格式化时间字符串	>>>time.ctime(1519652795.2386184) 'Mon Feb 26 21:46:35 2018'
time.sleep(seconds)	延迟 seconds 秒	>>>time.sleep(1) #将等待 1 秒

打开 Python Shell，尝试上方示例代码。

从上面的代码中可以看出，在 Python 里有三种方式表示时间，分别为时间截、时间元组、格式化时间字符串。

1. 时间戳

```
1519652795.2386184
```

时间戳是指北京时间 1970 年 01 月 01 日 08 时 00 分 00 秒起至现在这一刻的总秒数。

2. 时间元组

```
time.struct_time(tm_year=2018,tm_mon=2,tm_mday=26,tm_hour=21,tm_min=46,tm_sec=35,tm_wday=
0,tm_yday=57,tm_isdst=0)
```

元组中的 9 个元素如下表所示。

索引	属性	范围	值
0	tm_year（年）	>=1970	2018
1	tm_mon（月）	1～12	2
2	tm_mday（日）	1～31	26
3	tm_hour（时）	0～23	21
4	tm_min（分）	0～59	46
5	tm_sec（秒）	0～59	35
6	tm_wday（weekday）	0～6，周一是 0	0
7	tm_yday（一年中的第几天）	1～366	57
8	tm_isdst（是否是夏令时）	默认为-1	0

将时间元组存入变量 now 里之后，我们可以通过索引值找到具体的年、月、日、时、分、秒的数值。

```
>>>now = time.localtime(1519652795.2386184)
>>>now[0]
2018  #返回时间元组索引值为 0 的值
>>>print(str(now[0]) + "年" + str(now[1]) + "月" + str(now[2]) + "日")
2018 年 2 月 26 日
```

3. 格式化时间字符串

```
'Mon Feb 26 21:46:35 2018'
```

小贴士：时间戳与时区

时间戳是指格林尼治时间[UTC+0 时区]1970 年 01 月 01 日 00 时 00 分 00 秒（北京时间

[UTC+8 时区]1970 年 01 月 01 日 08 时 00 分 00 秒）起至现在的总秒数。

世界各地国家经度不同，时间也有所不同，通常每隔 15° 经度划分一个时区，共有 25 个时区。UTC 时区是经度为 0°的时区的时间，中国统一采用北京时间，经度为东经 120° 的时间。

由于地球自西向东自转，东边时间快于西边时间。因此，北京时间（ 120° ）比格林尼治时间（ 0° ）快 120 / 15 = 8 个小时。

下面练习一下时间模块的函数与属性的使用。

请编写一段代码，分别输出当前的年、月、日、时、分、秒、星期几、今年中的第几天。

实例

```
今年是 2018 年
本月是 2 月
今日是 26 号
此时是 21 点
此时是 46 分
此时是 35 秒
星期是周 1
是今年的第 57 天
```

打开 IDLE 编辑器，新建 Python 文件（ Shell 窗口：【 File 】→【 New File 】），保存好。

得到当前系统时间

1. 用 import 导入 time
2. 用 time.time()得到当前时间戳，用 time.localtime(timestamp)将时间戳转换为时间元组赋值给 timeTuple
3. 打印年
4. 打印月
5. 打印日期
6. 打印小时
7. 打印分
8. 打印秒
9. 打印星期，从 0 开始算，0 代表周一，所以要 + 1
10. 打印是今年第几天

```
1. import time
2. timeTuple = time.localtime(time.time())
```

```
3. print("今年是",timeTuple[0],"年")
4. print("本月是",timeTuple[1],"月")
5. print("今日是",timeTuple[2],"号")
6. print("此时是",timeTuple[3],"点")
7. print("此时是",timeTuple[4],"分")
8. print("此时是",timeTuple[5],"秒")
9. print("星期是周",timeTuple[6]+1)
10. print("是今年的第",timeTuple[7],"天")
```

点击【Run】→【Run Module】执行程序，看看是否能正确打印出当前时间。

9.4.2 日期时间模块

Python 日期时间（datetime）模块与时间模块相似，这个模块给予我们更多处理日期和时间的函数，用 import datetime 导入。datetime 的时间以对象形式表示，而 time 的时间戳则是一个浮点数。下面是 datetime 常见的函数。

函数	作用
myTime = datetime.datetime.now()	将这一刻的时间日期对象包括年、月、日、时、分、秒、微秒存到变量 myTime 中
myTime = datetime.datetime (2017,12,1,18,40,2)	将 2017 年 12 月 1 日 18 时 40 分 2 秒这个时间点存进一个变量 myTime 中
myTime.year	返回存在 myTime 里的时间点的年
myTime.month	返回存在 myTime 里的时间点的月
myTime.day	返回存在 myTime 里的时间点的日
myTime.hour	返回存在 myTime 里的时间点的时
myTime.minute	返回存在 myTime 里的时间点的分
myTime.second	返回存在 myTime 里的时间点的秒
myTime.microsecond	返回存在 myTime 里的时间点的毫秒

参考 9.4.1 节的练习，尝试使用 datetime 模块编写、输出当前时间的年、月、日、时、分、秒、毫秒的代码。

实例

```
今年是 2018 年
本月是 2 月
今日是 19 号
此时是 14 点
```

```
此时是 5 分
此时是 25 秒
此时是 836321 毫秒
```

参考答案见本书附赠资料。

9.5　webbrowser 模块

9.5.1　webbrowser 简介

Python 的 webbrowser 模块可以用 import webbrowser 导入，使用 webbrowser.open(url)能够帮我们打开这个 url 地址里的网页。

编写下面的代码：

```
>>>import webbrowser
>>>webbrowser.open("https://www.dreamcodestrue.com")
```

浏览器将自动打开网页 https://www.dreamcodestrue.com。

小贴士：URL

URL 全称是 Uniform Resource Locator，翻译为“统一资源定位符”。统一资源定位符标识了互联网上资源的位置和访问方法。

每个家庭都有地址，例如，小明家住在建国门外大街 150 号 701 室，在这个地址可以找到小明。而在互联网上，要找到不同的网页，也要用地址。这些地址就是 URL。

简而言之，可理解为互联网上资源的地址。互联网上的每个资源都有 URL。

9.5.2　案例 2：天气机器人

编写一个天气机器人，它将告诉我们某个城市的天气或雾霾情况。通过学习编写天气机器人，我们将练习 webbrowser 模块的使用，以及加深对 URL 的理解。

首先设想程序将询问用户要查询哪个城市，再问用户要查天气还是雾霾，最后用 webbrowser 打开要查询城市对应的网站页面。

实例

请输入要查的城市的拼音[北京:beijing]? >>>beijing
1-天气,2-雾霾 [1 或 2]? >>>2
#浏览器打开下面的页面:

本例要根据用户的输入,打开不同的 URL。例如，查询北京雾霾的 URL 就是 www.86pm25.com/city/beijing.html，让我们看看 URL 的组成。

打开浏览器，在浏览器的地址栏中输入下方的 URL，观察 URL 与网站内容，思考其中有什么规律?

http://www.86pm25.com/city/beijing.html
http://www.86pm25.com/city/shanghai.html
http://www.86pm25.com/city/shenzhen.html
http://www.86pm25.com/city/guangzhou.html

根据测试结果，上面网站打开的都是城市 PM2.5 的页面。通过观察，我们发现 URL 中的内容大多一样，唯一的区别在于城市的拼音。我们用"http://www.86pm25.com/city/" + 城市拼音字符串 + ".html"，即可组成查询任意城市雾霾的网页地址。

```
import webbrowser
city = input("请输入要查询的城市的拼音[北京:beijing]? >>>")
address = "http://www.86pm25.com/city/" + city + ".html"
```

```
webbrowser.open(address)
```

将 URL 字符串存入变量 address 里，再用 webbrowser.open(address)打开查询雾霾的网页。

同样，一些查询天气的网站 URL 也有类似规律。我们可以使用 webbrowser.open（"http://www.tianqi.com/" + 城市拼音字符串）打开查询天气的网址。

http:// www.tianqi.com/beijing
http:// www.tianqi.com/shanghai
http:// www.tianqi.com/shenzhen
http:// www.tianqi.com/shenyang

在理解了 URL 之后，我们可以尝试编写天气机器人了。打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

天气机器人

```
import webbrowser
while True:
→  city = input("请输入要查询的城市的拼音[北京:beijing]? >>>")
→  option = input("1-天气,2-雾霾  [1 或 2]? >>>")
→  if option == "1":
→  →  address = "https://www.tianqi.com/" + city
→  →  break
→  elif option == "2":
→  →  address = "http://www.86pm25.com/city/" + city + ".html"
→  →  break
→  else:
→  →  print("没听懂，请重新输入！")
→  →  continue  #关键词 continue 让我们进入下一次循环
webbrowser.open(address)
```

单击【Run】→【Run Module】执行程序，看看是否能正确调出城市的天气情况。

9.6　操作文件

9.6.1　操作系统的 os 模块

os 模块能够让我们用代码操作电脑系统，例如，可以新建文件夹、重命名文件夹等。模块用 import os 导入。下面是几个简单的 os 模块函数。

函数	作用	举例
os.getcwd()	获取当前工作路径	>>> os.getcwd() 'D:\\Python' 返回当前 Shell 的目录
os.mkdir(路径)	创建一个新的目录（文件夹）	>>> os.mkdir("C:\\hello") 在 C 盘新建一个名为 hello 的文件夹
os.chdir(路径)	直接切换到该路径，也就是该文件夹	>>> os.chdir("C:\\hello") 将目录切换到 C 盘的 hello 文件夹里
os.startfile(路径)	打开路径里的应用文件	>>> os.startfile("hi.txt") hi.txt 文件以记事本应用格式打开
os.rename(旧路径，新路径)	重命名文件	>>> os.rename("hi.txt" , "hello.txt") 将 hi.txt 文件重命名为 hello.txt
os.remove(路径)	删除该路径的文件	>>> os.remove("hi.txt") 删除 hi.txt

小贴士：绝对路径与相对路径

路径是文件夹或文件在计算机里的地址，分为绝对路径和相对路径。假设我的 C 盘里有一个 hello 文件夹，文件夹下有 hi.txt 文件和同学们的资料文件夹。

```
C
|_____hello
        |_____ hi.txt
        |_____ 同学们的资料
```

绝对路径是完整路径，从盘符开始标记，比如上面的记事本文档的绝对路径是 "C:\hello\hi.txt"。

相对路径则是相对于当前路径后的路径。假设我修改了当前路径为"C:\hello"。此时相对路径就是"hi.txt"，即"hi.txt"相对于当前路径"C:\hello"的路径。

如果需要查询文件路径，在 Windows 系统中可单击鼠标右键，在弹出的菜单中选择属性后，地址栏将给出当前文件夹的路径。在 Mac 系统中可单击鼠标右键，在弹出的菜单中选择显示简介后，地址栏将给出当前文件夹的路径。如需文件路径，则需加上文件名。

另外，在 Python 编程规范里，要把路径的一条斜杠换成两条斜杠。例如"C:\hello\hi.txt"应写成"C:\\hello\\hi.txt"。

在下面这段代码里，我们利用 os 模块，在“同学们的资料库”文件夹里，给五个学生自动建立了各自的文件夹。

```
import os
students = ["林美可", "方仲伊", "李涵欣", "陈蕾蕾", "韩盈"]
for student in students: #列表里有多少个学生，就用 for 循环重复多少次
→  os.mkdir("C:\\hello\\同学们的资料库\\" + student) #用路径和姓名字符串组成新路径
```

```
C
|_____hello
        |_____ hi.txt
        |_____ 同学们的资料
                    |_____ 林美可
                    |_____ 方仲伊
                    |_____ 李涵欣
                    |_____ 陈蕾蕾
                    |_____ 韩盈
```

不难想象，如果要给 2000 名学生或同事建立各自的文件夹，手动操作耗时长并容易出错。而用 Python 的 os 模块能够迅速完成任务。

9.6.2　案例 3：音乐倒计时

我想在电脑里给自己编写一个倒计时程序，设定好计时的分钟，倒计时结束后，放音乐提醒我去休息一下。通过这个项目，我们将练习 time、os、random 等模块的应用。

运行效果如下：

实例

```
你想学习多少分钟?>>>1
任务完成，听音乐休息一下吧
#响起几首我喜欢的音乐之一
```

打开 IDLE 编辑器，新建 Python 文件（Shell 窗口：【File】→【New File】），保存好。

音乐倒计时

1. 用 import 导入 os、time、random
2. 将多首音乐的绝对路径（修改为你自己的存储文件的地方），存入列表 songs 中。其中路径间隔符是\\，因为在字符串中转义字符才用\
3. 询问用户学习多少分钟，存入 mins
4. 计算出共有多少秒
5. 等待倒计时完成
6. 打印“听音乐休息”

```
7. 从 songs 中随机挑一首歌的路径，赋值给 song
8. 用 os 模块打开存在 song 里的路径，播放音乐

1. import os, time, random
2. songs = ["C:\\Users\\admin\\Music\\Beethoven.mp3",
            "C:\\Users\\admin\\Music\\Chopin.mp3",
            "C:\\Users\\admin\\Music\\cat-scream.wav"]

3. mins = int(input("你想学习多少分钟?>>>"))
4. secs = mins * 60
5. time.sleep(secs)
6. print("任务完成，听音乐休息一下吧")
7. song = random.choice(songs)
8. os.startfile(song)
```

点击【Run】→【Run Module】执行程序，看看是否能够按设计播放音乐。

9.6.3 案例 4：编写文档的 docx 模块

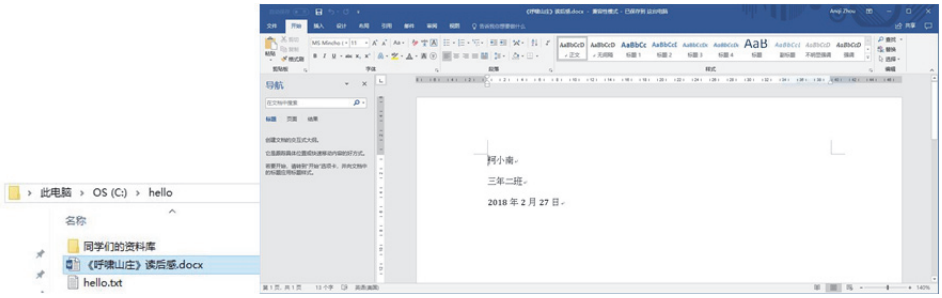
下面将通过一个例子介绍 docx 模块，用来新建 Word 文档，写入并保存。

docx 模块不是 Python 的内置模块，需使用 9.2.1 小节介绍的安装方法，用 pip install python-docx 指令安装 python—docx 模块。这个模块用 import docx 导入，注意它导入的名字和安装名字略有不同。

有的同学要定期写作业、写报告，每次开头的姓名、班级和日期都很类似，我们可以把这类任务自动化。设计的程序会询问用户作业名称，然后新建空白 Word 文档，自动插入姓名、班级、日期信息，以作业名为文件名保存，自动创建一个模板。

实例

```
你的作业叫什么? >>>《呼啸山庄》读后感
#新建 word 文档，文件名为《呼啸山庄》读后感，文件自带姓名、班级和日期
```



打开 IDLE 编辑器，新建 Python 文件（Shell 窗口:【File】→【New File】），保存好。

新建作业文档

1. 用 import 导入 os、docx、datetime
2. 询问作业名称
3. 设置路径为 C:\\Hello，进入目标文件夹
4. 用 docx.Document()新建一个文档存入变量 doc
5. 为 doc 添加一段“柯小南”
6. 为 doc 添加一段“三年二班”
7. 使用 datetime.datetime.now()找到现在的时间，存入变量 now 里
8. 用 now.year、now.month 和 now.day 组成日期字符串，添加进 doc 文档里
9. 修改文件名，添加扩展名.docx
10. 保存 doc，文件名为 title

```
1.  import os, docx, datetime
2.  title = input("你的作业叫什么？ >>>")
3.  os.chdir("C:\\Hello")
4.  doc = docx.Document()
5.  doc.add_paragraph('柯小南')
6.  doc.add_paragraph('三年二班')
7.  now = datetime.datetime.now()
8.  doc.add_paragraph(str(now.year) + "年" + str(now.month) + "月"+ str(now.day) + "日")
9.  title = title + ".docx"
10. doc.save(title)
```

点击【Run】→【Run Module】执行程序，看看是否能为三年级二班的柯小南自动创建作业文档模板。你也可以根据自己的需要，编写程序创建适合自己的 Word 文档模板。

9.7 总结及课后练习

这一章我们了解了几个 Python 常用的模块，包括 random、time、datetime、webbrowser、os、docx。有了这些模块，我们可以将 Python 跟其他程序和资源结合起来，在生活中真正开始应用。Python 还有许多有趣的模块，希望大家能大胆地尝试和探索，用编程解决身边的问题。

询问用户最喜欢的书名，用 webbrowser 模块打开豆瓣书籍搜索页面。

输入 https://www.douban.com/search?q=移动迷宫，将打开移动迷宫搜索页面。输入 https://www.douban.com/search?q=星球大战，将打开星球大战搜索页面。

第 10 章

Tkinter 界面——有按钮的软件

利用 Python 的 Tkinter 模块可以制作图形化用户界面，让我们的程序更美观、更好用。这一章我们将学习如何使用 Tkinter 模块制作传统意义上的软件程序。

10.1 GUI 与 CUI

之前我们编写的程序都是在 CUI（Command User Interface）上运行的，CUI 是命令行用户界面，即基于文字的界面。而大多数的软件是 GUI（Graphical User Interface），GUI 是图形用户界面，有按钮、下拉条等元素，看起来美观，用起来方便，如图 10-1 所示。

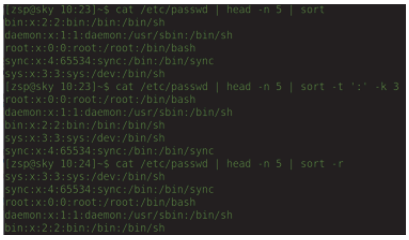
CUI（命令行用户界面）	GUI（图形用户界面）
	
早期 Linux 系统就是命令行用户界面 井字游戏的命令行用户界面	Windows 系统就是图形用户界面 井字游戏的图形用户界面
	

图 10-1　比较命令行用户界面和图形用户界面

本章将介绍如何运用 Tkinter 给 Python 程序添加 GUI 界面。

10.2 介绍 Tkinter 框架

利用 Python 的 Tkinter 模块可以制作图形化用户界面。我们可以通过 import tkinter 或 from tkinter import * 导入 Tkinter 模块。

实例

下面是 Tkinter 界面的一个例子，通过它可以了解 Tkinter 程序的框架，以及各部分的功能。这是一个单一按钮的程序，按一下按钮将改变它的背景颜色，如图 10-2 所示。

图 10-2 彩色按钮程序

导入相关的模块	`from tkinter import *` `import random`
新建窗体，存入变量 window 中	`window = Tk()`

设置窗体内的程序，包括有哪些控件、控件长什么样（控件属性）、设置每个控件要干什么（程序逻辑）	`myButton = Button(window,text="换颜色")`
	`myButton.pack()`
	
	`def change(event):`
	`  colors = ["red","orange","yellow","green","cyan"]`
	`  myButton.configure(bg = random.choice(colors))`
	
	`myButton.bind("<Button-1>",change)`

运行窗体程序	`window.mainloop()`

一般的 Tkinter 程序可以分为四部分：导入模块、设置窗口（我们称为“窗体”）、设置窗体内的程序和运行窗体程序。

导入模块、设置窗口和运行窗体程序比较简单，这里主要介绍设置窗体内的程序，包括可以将哪些控件放进窗体里、如何设置它们，以及如何给它们编写动作程序。

10.3　给窗体添加控件

建立好窗体之后，我们来学习如何设置窗体里的控件。首先看一下可以添加哪些控件。下表列举了控件及其含义，我们会详细介绍几个常用控件的程序。

控件	描述
Button	按钮控件，在程序中显示按钮
Label	标签控件，可以显示文本和位图
Entry	输入控件，用于输入文本内容
Scrollbar	滚动条控件，用于翻页
Canvas	画布控件，显示图形元素如线条或文本
Radiobutton	单选按钮控件，显示单选的按钮状态
Checkbutton	多选框控件，用于在程序中提供多项选择框
Spinbox	输入控件，与 Entry 类似，但是可以指定输入范围值
Menu	菜单控件，显示菜单栏、下拉菜单和弹出菜单
Menubutton	菜单按钮控件，用于显示菜单项
OptionMenu	选择菜单控件
Listbox	列表框控件，Listbox 窗体小部件用于显示一个字符串列表
Text	文本控件，用于显示多行文本
Message	消息控件，用来显示多行文本，与 Label 比较类似
tkMessageBox	用于显示应用程序的消息框
Scale	范围控件，显示一个数值刻度，为输出限定范围的数字区间
Frame	框架控件，在屏幕上显示一个矩形区域，用来作为容器
Toplevel	容器控件，用来提供一个单独的对话框，和 Frame 类似
LabelFrame	LabelFrame 是一个简单的容器控件，常用于复杂的窗体布局
PanedWindow	窗体布局管理的插件，可以包含一个或者多个子控件

知道了上面的控件及其含义，我们如何将控件添加到窗体里呢？

```
from tkinter import *
window = TK()

#设置窗体内的控件
控件变量名称 = 控件(父容器,属性)
控件变量名称.pack()

window.mainloop()
```

```
from tkinter import *
window = Tk()

#设置窗体内的控件
myButton = Button(window, text="点击我吧")
myButton.pack()

window.mainloop()
```

在右边的例子里，我们生成了一个按钮，将它放在父容器 window（窗体）里，设置它的显示文字为“点击我吧”。生成按钮后，将它的 ID 存在 myButton 变量里，这样之后只要使用 myButton 就能找到对应的按钮。最后我们用 pack()将它在窗体里安排好。

所有的控件都可以采用上面的方法创建。新建一个控件对象，设置好它的父容器及属性，包括该控件的宽、高、背景颜色、文本等，再用 pack()布局。创建好控件后，要告诉计算机将控件放在父容器的哪个位置。pack()就是指自动缩放调整到合适的大小位置。控件放置的顺序根据 pack()代码顺序决定。

下面我们将给出几个控件的例子。

我们可以在页面上添加一个**按钮（Button）**，让用户点击，如图 10-3 所示。

添加按钮

```
from tkinter import *
window = Tk()

myButton = Button(window,text="点击我吧")
myButton.pack()

window.mainloop()
```

图 10-3　简单的按钮

标签（Label）可以显示文字。Message 和 Text 也可以用来显示文字，不过它们常用于显示多行文字，其中，使用 Text 可显示多种字体样式。下面是一个添加标签的代码，效果如图 10-4 所示。

添加标签

```
from tkinter import *
window = Tk()

myLabel = Label(window,text="欢迎来到梦想编程现实")
myLabel.pack()

window.mainloop()
```

图 10-4　简单的标签

文本框（Entry）可以让用户输入信息，给出一个空白的输入框，如图 10-5 所示。

添加文本输入框

```
from tkinter import *
window = Tk()
```

```
myEntry = Entry(window)
myEntry.pack()

window.mainloop()
```

图 10–5　简单的文本框

文本框带有两个非常有用的函数，帮助读取和修改文本框的值。

```
myEntry.get() #读取文本框里的内容
myEntry.insert(0,"Python") #从开头（索引值为 0）开始插入第二个参数里的文字
```

画布（Canvas）可以让我们绘制线条、图案、图片，也可以用来制作动画和游戏。画布的功能十分强大，我们将在 10.7 节里详细介绍它的功能。下面是添加画布的代码，图 10–6 展示了一个简单的空画布。

添加画布

```
from tkinter import *
window = Tk()

myCanvas = Canvas(window)
myCanvas.pack()

window.mainloop()
```

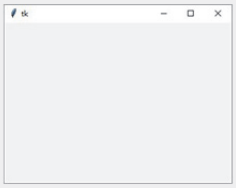


图 10–6　简单的画布

10.4　让控件变漂亮

10.4.1　为控件设置属性的方法

仅仅添加控件，使程序看上去十分单调。我们可以通过设置这些控件的属性来改变它们的颜色、字体、大小等。对窗体里控件属性的设置有两种方法，如下所示。

```
控件变量名称 = 控件(父容器,属性 1=值,属性 2=值,属性 3=值…) #方法 1
控件变量名称["属性"] = 值 #方法 2
控件变量名称.pack()
```

第一种方法，在创建控件时设置属性值。此时 pack()可以跟在创建控件的后面。

设置属性值的方法 1

```
from tkinter import *
window = Tk()
```

```
myLabel = Label(window, text = "梦想编程现实", fg = "lightblue", bg = "white").pack()

window.mainloop()
```

在上面的例子中，fg 是指 foreground 前景色，也就是字体颜色，而 bg 是指 background 背景色。设置这两个属性可以改变控件的颜色，如图 10-7 所示。

图 10-7　有颜色的标签

第二种方法，在创建控件后，通过键值对的方式设置属性值，如 myButton["bg"] = "pink"，类似字典数据类型。使用这种方法 pack()不可以直接跟在创建控件的后面，而要等所有属性设置完成后才可以使用 pack()，参考下面的代码及图 10-8。

设置属性值的方法 2

```
from tkinter import *
window = Tk()

myButton = Button(window,text="点击一下")
myButton["bg"] = "pink"
myButton["fg"] = "white"
myButton["height"] = 1
myButton["width"] = 12
myButton.pack()

window.mainloop()
```

图 10-8　有颜色的按钮

了解了属性的用法后，我们看看各控件的常用属性。

10.4.2　控件的常用属性

下表为控件的常用属性及描述、示例。

属性	描述	示例
text	指定按钮上显示的文本	text = "hello"
font	指定按钮上文本的字体	font = "宋体 16 bold italic"
foreground(或 fg)	指定按钮的前景色	fg = "red"
background 或 bg	指定按钮的背景色	bg = "black"
borderwidth 或 bd	指定按钮边框的宽度	bd = "12px"

续表

属性	描述	示例
width	指定按钮的宽度	width = 10
height	指定按钮的高度	height = 2
state	指定按钮的状态，disabled 表示停止使用	state = "disabled"
activeforeground	按下时前景色	activeforeground = "red"
activebackground	按下时背景色	activebackground = "black"
command	指定按钮被按时调用函数，值为函数名	command = greeting

编写下面的代码，尝试为按钮设置多种不同的属性，包括边框、按下按钮时的背景颜色和按下按钮时调用的函数，效果如图 10-9 所示。

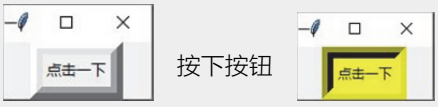
尝试按钮的多种属性

```
from tkinter import *
window = Tk()

def hello():
→  print("hello")

myButton = Button(window,text="点击一下")
myButton["bd"] = "10px" #较粗的边框
myButton["activebackground"] = "yellow" #按下按钮时背景为黄色
myButton["command"] = hello #按下按钮将调用 hello 函数
myButton.pack()

window.mainloop()
```



10-9　带属性按钮

标签的属性与按钮的属性类似，可以改变大小、字体和颜色。输入框 Entry 有个特别的属性 show，如果 show 为 False，那么不论输入什么文字都将显示 0，常用于隐藏密码，如图 10-10 所示。字体（font）属性的设置格式是“字体、字号、加粗、倾斜”，可以全部使用，也可以部分使用。在下面的例子中，我们仅设置了字体和字号。

尝试标签和输入框的多种属性

```
from tkinter import *
window = Tk()

myLabel = Label(window,text = "请输入密码",font = "宋体 16")
myLabel.pack()

myEntry = Entry(window,show = False,width = 20)
myEntry.pack()

myButton = Button(window,text = "提交")
myButton.pack()

window.mainloop()
```

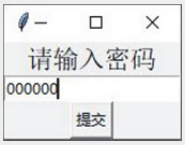


图 10-10　不显示内容的输入框

10.4.3　使用 config 配置属性

创建完控件后，我们也可以使用 config 重新配置控件的属性。在下面的例子里，我们用 config 改变了字体和显示的文字，效果如图 10-11 所示。

用 config 设置标签的字体

```
from tkinter import *

window = Tk()

myLabel = Label(window,text="我喜欢编程")
myLabel.pack()

myLabel.config(font="宋体 16 bold italic",text="我还是喜欢编程")
#改变字体和显示的文字

window.mainloop()
```

图 10–11　使用 config 函数改变属性

10.5　让窗体里的东西动起来

在前面我们学习了创建控件及改变控件的样式。我们仅仅是将控件拿出来摆在窗体中，但操作控件时没有任何效果。如果我们希望操作控件有效果，那么就要用到事件。

以 10.2 节里的彩色按钮为例。当按钮被点击时，按钮的背景颜色将改成任意颜色，如图 10–12 所示。

图 10–12　彩色按钮程序

这里的代码逻辑分为两个部分，一个是鼠标点击事件，另一个是颜色改变的函数。我们用代码将这两个部分在按钮 myButton 上绑定，所以每次 myButton 被鼠标点击时，就会调用 change 函数，改变按钮的背景颜色。

	`from tkinter import *`
	`import random`
	
	`window = Tk()`
	
	`myButton = Button(window,text="换颜色")`
	`myButton.pack()`

定义 change 函数，改变按钮背景颜色	`def change(event):`
	`→ colors = ["red","orange","yellow","green","cyan"]`
	`→ myButton.configure(bg = random.choice(colors))`

在按钮上绑定点击事件和 change 函数	`myButton.bind("<Button-1>",change)`
	`window.mainloop()`

下面总结了三种方法绑定控件和做事情的函数。其中，函数必须要带有一个 event 的参数，方便绑定控件上的事件和这个函数。

```
def 函数名:
    #要做的事情
```

第一种方法如上面的例子，在某个控件上绑定它的事件和函数。当事件在它身上发生时，即调用被绑定的函数。

```
控件变量名称.bind(什么事件,函数名)
```

第二种方法 bind_all 是全程序级别的绑定，只要在这个窗体的任何控件上发生这个事件就将调用被绑定的函数。

```
控件变量名称.bind_all(什么事件,函数名)
```

第三种方法直接在 command 属性中给出函数名，适用于按钮。

```
控件变量名称 = 控件(父容器,command=函数名)
```

下面我们来看看具体的事件，包括鼠标事件、键盘事件、窗体事件等。学会这些事件后，我们就可以开始灵活编写各种软件和游戏。

10.5.1 Tkinter 里的事件

下表列出了各类鼠标事件，其中 1 代表鼠标左键，2 代表鼠标中键，3 代表鼠标右键，如图 10-13 所示。如果鼠标的形状和标准的形状有出入，那么需要同学们测试一下每个数字对应的键。

事件	描述
<Button-1>	单击鼠标左键
<ButtonPress-1>	按下鼠标左键
<ButtonRelease-1>	释放鼠标左键
<B1-Motion>	按住鼠标左键移动
<Double-Button-1>	双击鼠标左键
<Enter>	鼠标指针进入某一控件区域
<Leave>	鼠标指针离开某一控件区域
<MouseWheel>	滚动滚轮

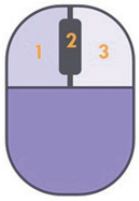


图 10-13 Tkinter 里的鼠标事件

练习下面的编程案例，绑定左键点击事件和添加新标签的函数，每次点击按钮，就会将“啥事？”加入窗口，效果如图 10-14 所示。

点击按钮添加标签

```
from tkinter import *
window = Tk()
```

```
myButton = Button(window, text="点击我吧!")
myButton.pack()
def addLabel(event):
→  global window #找到全局变量 window 对应的窗体
→  Label(window, text="啥事? ").pack() #新建标签
myButton.bind("<Button-1>", addLabel) #绑定左键点击和 addLabel 函数

window.mainloop()
```

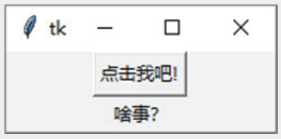


图 10-14　点击按钮添加标签

除鼠标事件外，还有下面的键盘事件可以使用。

事件	描述
<KeyPress-A>	按下 A 键，A 键可用其他键替代
<Alt-KeyPress-A>	同时按下 Alt 键和 A 键；Alt 键可用 Ctrl 键和 Shift 键替代
<Double-KeyPress-A>	快速按两下 A 键
<Lock-KeyPress-A>	大写状态下按 A 键
<Key>	按下任意键

10.5.2　响应事件中的属性

每个事件发生时，都有相关的信息存在 event 对象的属性里，比如对于按键事件，究竟用户按下了哪个键？对于鼠标事件来说，用户在哪个位置单击了鼠标？这些信息将帮助我们编写游戏和程序，我们可以通过 event.属性调用这些信息。

属性	描述
char	按键字符，仅对键盘事件有效，返回按键对应的字符
num	鼠标按键，仅对鼠标事件有效，返回单击的鼠标键对应的数字
type	所触发的事件类型
widget	引起事件的控件
x, y	鼠标当前位置，相对于控件
x_root, y_root	鼠标当前位置，相对于整个窗口

通过一个例子探索如何使用事件中的属性。

创建一个按钮，点击一次按钮添加一个标签，输出点击的位置，包括点击事件在按钮上的位置及在窗体上的位置，效果如图 10-15 所示。

点击事件在窗体里的位置

```
from tkinter import *
window = Tk()

myButton = Button(window, text="点击我吧!")
myButton.pack()
def addLabel(event):
    global window
    s = "你点的是按钮上X:", event.x, "Y:",event.y, "，窗体上X:", event.x_root, "Y:",event.y_root, "哟~"
    Label(window,text=s).pack()
myButton.bind("<Button-1>",addLabel)

window.mainloop()
```

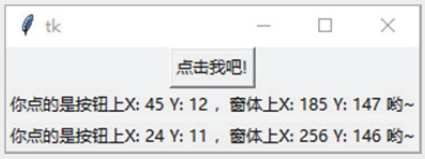


图 10-15　使用事件中的属性练习示例

小贴士：X、Y坐标

在程序的世界里，我们通常用 X、Y坐标在二维平面上定位。X坐标决定了一个点的水平位置，Y坐标决定了一个点的垂直位置。如图 10–16 所示，三角形的水平 X坐标为 50，垂直 Y坐标为 30。

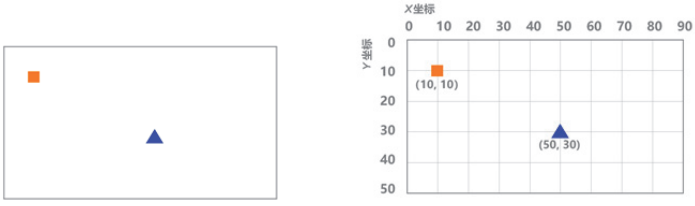


图 10–16　Tkinter 的二维坐标

左上角顶点的坐标是(0,0)，水平向右 X坐标不断增加，垂直向下 Y坐标不断增加。

在 Tkinter 的点击事件里有两种坐标。event.x 和 event.y 是相对坐标，该点坐标以控件左上角为基准。event.x_root 和 event.y_root 是绝对坐标，该点坐标以窗体左上角为基准。

10.6　案例 1：绘图软件

本节将制作一款完整的绘图软件。我们将使用画布 Canvas 控件绘图，并练习 Tkinter 里的事件绑定。

我们先从简单的绘图软件版本开始，并在后面的章节里进行改进。

在第一代绘图软件里，用户将使用三种不同的颜色在画布上用鼠标绘画，如图 10-17 所示。

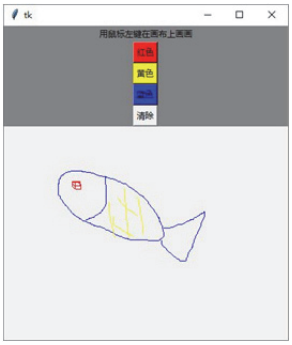


图 10-17　第一代绘图软件展示

在编写绘图软件之前，我们先详细了解一下画布。画布的主要属性与 10.4 节里学习的属性类似，我们可以通过属性修改画布的大小及背景颜色，而画布的函数则有许多。

例如，下面的 create_line 函数会在 myCanvas 画布上画一条线段，线段的两个端点分别为(10,10)和(200,300)。我们将这个线段的 ID 存入名为 shape1 的变量内。

```
shape1 = myCanvas.create_line(10,10,200,300)
```

下面列举了其他画布的函数，希望大家逐一进行练习。

方法	描述	举例
create_line	画线	myCanvas.create_line(10,10,50,50,fill = "yellow") 将在(10,10)，(50,50)两点之间画黄色的线段
create_rectangle	绘制矩形(a,b,c,d),值为左上角和右下角的坐标	myCanvas.create_rectangle(10,10,110,110) 将在(10,10)，(110,110)两点之间画矩形
create_oval	画圆	myCanvas.create_oval(10,10,200,300) 将在(10,10)，(200,300)两点间的矩形内画圆或椭圆
create_text	绘制文字	myCanvas.create_text((10,10),text = 'Hello') 将在(10,10)点绘制 Hello 文字
delete	删除绘制的图形	myCanvas.delete(shape1) 将删除存在 shape1 变量里的图形

续表

方法	描述	举例
move	移动图像(图像,*x* 横移,*y* 纵移)。然后用 window.update()刷新即可看到图像的移动	myCanvas(shape1, 20, -10) 将移动 shape1，往右移动 20，往上移动 10
coords	返回对象的位置的两个坐标（4 个浮点数的列表）	print(myCanvas.coords(shape1)) [100.0, 200.0, 400.0, 300.0] 将得到 shape1 的坐标列表

10.6.1 制作绘图软件 1

下面我们就来编写第一代绘图软件，操作主要分为以下四步。

（1）添加所有页面所需的控件，包括标签、按钮和画布等，如图 10-18 所示。

```
from tkinter import *
window = Tk()

window.configure(background="gray")

instruction = Label(window,text="用鼠标左键在画布上画画",
background="gray")
instruction.pack() #创建标签

redButton = Button(window,text="红色",bg="red")
redButton.pack() #创建红色按钮

yellowButton = Button(window,text="黄色",bg="yellow")
yellowButton.pack()#创建黄色按钮

blueButton = Button(window,text="蓝色",bg="blue")
blueButton.pack() #创建蓝色按钮

clearButton = Button(window,text="清除")
clearButton.pack() #创建清除按钮

myCanvas = Canvas(window,width=400,height=300)
myCanvas.pack() #创建画布

#改变画笔颜色的程序
```

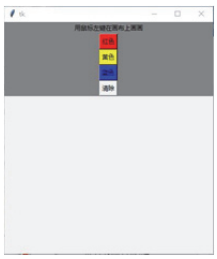


图 10-18 第一代绘图软件控件

```
#清除画布的程序
#画线的程序

window.mainloop()
```

（2）编写改变画笔颜色的相关程序。用一个变量 myColor 来存储当前的画笔颜色，并编写改变 myColor 的函数，将函数与相关按钮的点击事件绑定。

```
#改变画笔颜色的程序
myColor = "black"

def set_red(event): #改 myColor 为红色的函数
    global myColor
    myColor = "red"
redButton.bind("<Button-1>",set_red) #当按下红色按钮，设置画笔颜色为红色

def set_blue(event): #改 myColor 为蓝色的函数
    global myColor
    myColor = "blue"
blueButton.bind("<Button-1>",set_blue) #当按下蓝色按钮，设置画笔颜色为蓝色

def set_yellow(event): #改 myColor 为黄色的函数
    global myColor
    myColor = "yellow"
yellowButton.bind("<Button-1>",set_yellow) #当按下黄色按钮，设置画笔颜色为黄色
```

（3）编写清除画布的程序。定义一个 clear 函数，用 myCanvas.delete(ALL)清空画布上的所有图形，将函数与相关的按钮绑定。

```
#清除画布的程序
def clear(event):
    myCanvas.delete(ALL)
clearButton.bind("<Button-1>",clear)
```

（4）编写画线的程序。

在画线的程序中有以下两个主要操作。

当用户按下鼠标左键时，用全局变量 prevX 和 prevY 记录当前点的 X 坐标和 Y 坐标，作为画线的起点。这个步骤将会存储在 pen_down 函数里。我们将记录当前点的 pen_down 函数与在画布上按下鼠标左键的事件绑定。

当用户按下鼠标左键移动时，首先从前点（prevX,prevY）到当前点（event.x,event.y）画

线。再把当前点存储到前点信息变量 prevX 和 prevY 里，为画下一条线做准备。每当鼠标移动一小下就画一条小线段，多条小线段加在一起组成流畅的线条。这个画线的步骤将会存储在 draw 函数里。我们将这个画线的函数 draw 和移动鼠标左键的事件进行绑定。

```
#画线的程序
def pen_down(event):
    global prevX
    global prevY
    prevX = event.x
    prevY = event.y
myCanvas.bind("<ButtonPress-1>",pen_down)

def draw(event):
    global prevX
    global prevY
    myCanvas.create_line(prevX,prevY,event.x,event.y,fill=myColor)
    prevX = event.x
    prevY = event.y
myCanvas.bind("<B1-Motion>",draw)
```

这样，第一代绘图软件就完成了，轮到你来自己尝试制作了。下面是完整的代码，可以参考。

绘图软件 1

```
from tkinter import *
window = Tk()

window.configure(background="gray")

instruction = Label(window,text="用鼠标左键在画布上画画",background= "gray")
instruction.pack() #创建标签并放置

redButton = Button(window,text="红色",bg="red")
redButton.pack() #创建红色按钮

yellowButton = Button(window,text="黄色",bg="yellow")
yellowButton.pack() #创建黄色按钮

blueButton = Button(window,text="蓝色",bg="blue")
blueButton.pack() #创建蓝色按钮

clearButton = Button(window,text="清除")
```

```
clearButton.pack() #创建清除按钮

myCanvas = Canvas(window,width=400,height=300)
myCanvas.pack() #创建宽 400、高 300 的画布


#改变画笔颜色的程序
myColor = "black"

def set_red(event):
→   global myColor
→   myColor = "red"
redButton.bind("<Button-1>",set_red)

def set_blue(event):
→   global myColor
→   myColor = "blue"
blueButton.bind("<Button-1>",set_blue)

def set_yellow(event):
→   global myColor
→   myColor = "yellow"
yellowButton.bind("<Button-1>",set_yellow)

#清除画布的程序
def clear(event):
→   myCanvas.delete(ALL)
clearButton.bind("<Button-1>",clear)

#画线的程序
def pen_down(event):
→   global prevX
→   global prevY
→   prevX = event.x
→   prevY = event.y

myCanvas.bind("<ButtonPress-1>",pen_down)


def draw(event):
→   global prevX
```

```
    global prevY
    myCanvas.create_line(prevX,prevY,event.x,event.y,fill=myColor)

    prevX = event.x
    prevY = event.y

myCanvas.bind("<B1-Motion>",draw)

window.mainloop()
```

10.6.2 制作绘图软件 2

只能用四种颜色画画太受限了，我们可以添加 Tkinter 的 colorchooser 模块，调用程序里的调色盘，让颜色选择更加广泛，如图 10-19 所示。我们用新的程序替换改变画笔颜色程序的部分内容。

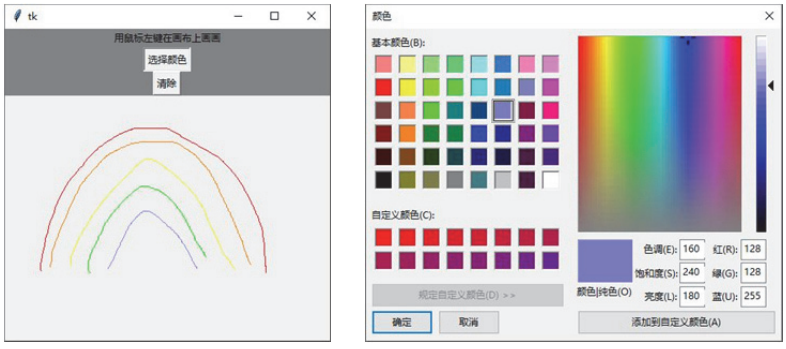


图 10-19 绘图软件第二代展示

Tkinter 的 colorchooser 模块帮助我们调用程序的调色盘。通过 askcolor()调用，函数 askcolor()将返回一个元组，如((0,0,255),"#0000a0")，以两种不同的方式表达颜色。我们将用索引值 1 选择第二种颜色表达式。

在第二代程序里，我们将之前的颜色按钮替换成一个选择颜色按钮，定义一个 pick_color 函数，将新的颜色表达式赋值给 myColor 变量。

```
#改变画笔颜色的程序
from tkinter.colorchooser import *

colorButton = Button(window,text="选择颜色")
colorButton.pack()
```

```
myColor = "black"

def pick_color(event):
→  global myColor
→  myColor = askcolor()
→  myColor = myColor[1]
colorButton.bind("<Button-1>",pick_color)
```

下面是完整的代码。

绘图软件 2

```
from tkinter import *
from tkinter.colorchooser import *

window = Tk()
window.configure(background="gray")

instruction = Label(window,text="用鼠标左键在画布上画画",background= "gray")
instruction.pack() #创建标签并放置

colorButton = Button(window,text="选择颜色")
colorButton.pack() #创建改变颜色的按钮

clearButton = Button(window,text="清除")
clearButton.pack() #创建清除按钮

myCanvas = Canvas(window,width=400,height=300)
myCanvas.pack() #创建宽 400、高 300 的画布


#改变画笔颜色的程序
myColor = "black"

def pick_color(event):
→  global myColor
→  myColor = askcolor()
→  myColor = myColor[1]
colorButton.bind("<Button-1>",pick_color)


#清除画布的程序
```

```
def clear(event):
    myCanvas.delete(ALL)
clearButton.bind("<Button-1>",clear)


# 画线的程序
def pen_down(event):
    global prevX
    global prevY
    prevX = event.x
    prevY = event.y

myCanvas.bind("<ButtonPress-1>",pen_down)


def draw(event):
    global prevX
    global prevY
    myCanvas.create_line(prevX,prevY,event.x,event.y,fill=myColor)

    prevX = event.x
    prevY = event.y

myCanvas.bind("<B1-Motion>",draw)

window.mainloop()
```

编写好后试一下吧，看看是否可以成功调用调色盘。

10.6.3 制作绘图软件 3

仅仅画线条还不够有趣，我们可以改进程序，让它能够画圆形和方形，如图 10-20 所示。我们将使用 myShape 存储要画的形状，并用条件判断语句决定画布所创造的图形。

在程序里我们会用到画布的两个函数 create_oval()和 create_rectangle()。函数的具体用法如下。

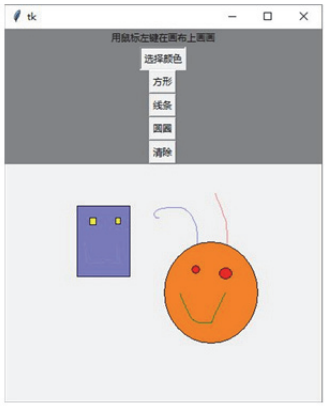


图 10-20 绘图软件第三代展示

```
myCanvas.create_oval(prevX,prevY,event.x,event.y,fill=myColor)
```

这行代码将在鼠标前点（prevX,prevY）和当前点（event.x,event.y）间画一个椭圆形，然后用画笔颜色填充。

```
myCanvas.create_rectangle(prevX,prevY,event.x,event.y,fill=myColor)
```

在鼠标前点（prevX,prevY）和当前点（event.x,event.y）间画一个矩形，然后用画笔颜色填充。

下面是具体的代码。

绘图软件 3

```
from tkinter import *
from tkinter.colorchooser import *

window = Tk()
window.configure(background="gray")

instruction = Label(window,text="用鼠标左键在画布上画画",background= "gray")
instruction.pack()

colorButton = Button(window,text="选择颜色")
colorButton.pack()

rectButton = Button(window,text="方形")
rectButton.pack()

lineButton = Button(window,text="线条")
lineButton.pack()

circleButton = Button(window,text="圆圈")
circleButton.pack()

clearButton = Button(window,text="清除")
clearButton.pack()

myCanvas = Canvas(window,width=400,height=300)
myCanvas.pack()

myShape = "line" #使用 myShape 变量存储当前绘画的图形
myColor = "black"
```

```
def pen_down(event):
    global prevX
    global prevY
    prevX = event.x
    prevY = event.y
myCanvas.bind("<ButtonPress-1>",pen_down)
#按下鼠标左键时，得到起始点的 X、Y 坐标

def draw(event):
    global prevX
    global prevY
    if myShape == "line":
        myCanvas.create_line(prevX,prevY,event.x,event.y,fill=myColor)
        prevX = event.x
        prevY = event.y
myCanvas.bind("<B1-Motion>",draw) #当鼠标移动时，如果选择画线，就画线

def pen_up(event):
    if myShape == "circle":
        myCanvas.create_oval(prevX,prevY,event.x,event.y,fill=myColor)
    if myShape == "rectangle":
        myCanvas.create_rectangle(prevX,prevY,event.x,event.y,fill=myColor)
myCanvas.bind("<ButtonRelease-1>",pen_up)
#松开鼠标左键时，如果 myShape 里是圆形，就绘制圆形。如果 myShape 里是矩形，就绘制矩形

def pick_color(event):
    global myColor
    myColor = askcolor()
    myColor = myColor[1]
colorButton.bind("<Button-1>",pick_color)

def clear(event):
    myCanvas.delete(ALL)
clearButton.bind("<Button-1>",clear)

def set_line(event):
    global myShape
    myShape = "line"
lineButton.bind("<Button-1>",set_line)
#点击线条按钮时，设 myShape 为线条"line"
```

```
def set_rect(event):
    global myShape
    myShape = "rectangle"
rectButton.bind("<Button-1>",set_rect)
#点击方形按钮时，设 myShape 为矩形"rectangle"

def set_circle(event):
    global myShape
    myShape = "circle"
circleButton.bind("<Button-1>",set_circle)
#点击圆形按钮时，设 myShape 为圆形"circle"

window.mainloop()
```

除绘制不同的形状外，我们还可以选择改变边框的颜色、线条的粗细等。希望大家多多尝试，不断完善自己的程序。

10.7　案例 2：编写桌面备忘录

我希望编写一个备忘录，并固定在桌面上，提醒我该做的事情，如图 10-21 所示。在下面这个编程挑战练习里，我们给出了一串简单的代码，创建固定在计算机桌面的备忘录并编写程序，让用户可以通过双击鼠标左键关闭备忘录。

图 10-21　固定桌面的备忘录展示

固定在电脑桌面的备忘录

1. 导入 Tkinter 模块
2. 创建窗体
3. 设置窗体大小与位置
4. 去除窗体边框
5. 设置窗体透明度
6. 创建文本域，背景设为粉色
7. 插入文本“12:00 吃饭”
8. 放置文本域控件
9. 定义关闭窗体函数
   1）声明全局变量 window
   2）关闭窗体
10. 窗体绑定鼠标左键双击
11. 运行窗体

```
from tkinter import *
window = Tk()
window.geometry("120x150+1100+50")
window.overrideredirect(True)
window.attributes("-alpha", 0.5)
txt = Text(window,bg = "pink")
txt.insert(INSERT,"12:00 吃饭")
txt.pack()
def closeWin(event):
    global window
    window.destroy()

window.bind("<Double-Button-1>",closeWin)
window.mainloop()
```

编写好代码后运行一下。

你还可以根据自己工作和学习的需要，给自己编写更多的程序，通过编程让自己的生活更加智能和高效。

10.8 总结及课后练习

在这一章里，我们学习了如何用 Tkinter 编写图形用户界面，包括如何添加控件、如何设置控件的属性、如何绑定事件和动作。希望大家在本章学习结束后，能够大胆尝试，创造出更多有图形用户界面的程序，从而帮助我们更高效地生活和工作。

编写骰子：设置一个正方形的按钮，点击该按钮后，按钮上的文本会变为 1～6 中的任意一个数。

第 11 章

密码的奥妙——众目睽睽之下的悄悄话

本章将通过 Python 介绍密码学，伴随故事背景介绍不同的加密方法，并利用前面所学的知识，用 Python 将加密程序编写出来。

11.1 打胜仗要靠算法

凯撒大帝是杰出的军事家，为后来的罗马帝国开疆拓土，打了无数胜仗。他在行军时经常要给士兵们传达指令，而万一这些指令被敌人的间谍截获，敌人获悉了他们的计划，打仗就非常被动了。

公元前 58 年左右，凯撒大帝想出了一个办法，他决定把自己的指令加密。他告诉自己的士兵们，出征后，指令的每个字母都会往右移 3 格。这样就算敌人截获了信息，看到的也是无法理解的一堆字母。这是流传下来的早期加密方法。

此后，但凡想要传递不被别人知道的信息，大家就开始用密码。大家约定好一个规则，给明文加密，即把明文变成密文传递。收到密文后，根据规则解密就可以转回明文。

只要学好逻辑学和数学，早期密码还是比较容易破译的。

到了第二次世界大战时，纳粹德国发明了 Enigma，即可以用机器加密，其加密速度和复杂度都比之前的加密方法优越很多。这个加密方法成了希特勒的重要武器。

英国外交部通信处召集了包括“计算机科学之父”艾伦 · 图灵（Alan Turing）在内的多位计算机科学家、数学家、语言学家，经过艰难的工作，最终成功设计出能够破解 Enigma 的机器，帮助盟军取得了“二战”的

胜利。

除军事用途外，通信行业也使用编码，比如早期的电报使用的就是莫尔斯码。如今计算机和互联网渗透到了我们生活中的方方面面，如何保护好银行卡、网络账户的密码等重要信息，是现代密码学的关键。

在这一章里，我们会了解不同的加密方法，并编写加密的程序。

11.2 案例 1：倒着说话——调转密码

11.2.1 调转密码介绍

首先，我们来看一种简单的加密方法。调转密码通过把文字倒着打印加密。

明文	意思	密文
i love apples	我喜欢苹果	selppa evol i
attack tomorrow	明日进攻	worromot kcatta

这种加密方法的加密程度非常弱。例如，“吧了么什说想我道知以可就，看一看要只你”。因为它比较简单，我们就当成是热身练习吧。

11.2.2 编写调转密码

编写一段代码，让用户输入明文，计算出调转密码的密文后，打印出密文。

实例

```
请输入明文>>>i love apples
你的密文是： selppa evol i
```

调转密码程序 1

1. 定义函数 encrypt，带参数 original，这个参数是明文的字符串
   1）original[::-1]将 original 里明文的字符串反转，将返回的结果存到局部变量 code 里
   2）函数 encrypt 返回存在局部变量 code 里的密文
2. 用 input()让用户输入明文，将明文字符串存到变量 text 里
3. 调用 encrypt 函数，将 text 变量里的明文字符串作为参数，将返回的密文值存到变量 secret 里
4. 打印密文

```
1. def encrypt(original):
   →  code = original[::-1]
```

```
    return code

text = input("请输入明文>>>")
secret = encrypt(text)
print("你的密文是：", secret)
```

在第 4 章里，我们介绍了如何处理字符串。例如：

c = "吃葡萄不吐葡萄皮"
c[1 : 7] 会返回"葡萄不吐葡萄"
c[1 : 7 : 2] 会返回"葡不葡"
c[7 : 1 : -1] 会返回"皮萄葡吐不萄"
c[: : -1] 会返回"皮萄葡吐不萄葡吃"

编写好后保存，运行代码。看看如果输入 attack tomorrow，是否可以得到下面的结果？

```
请输入明文>>> attack tomorrow
你的密文是：  worromot kcatta
```

11.2.3　编写调转密码窗口

为了改进程序，我们可以利用前面学的 Tkinter GUI 库，编写一个有文本框、按钮和标签的小窗口，方便大家使用，如图 11-1 所示。

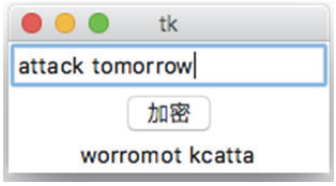


图 11-1　调转密码窗口

调转密码程序 2

1. 导入 Tkinter 模块
2. 设置窗口
3. 在窗口里生成一个文本框，命名为 inputBox
4. 在窗口里生成一个按钮，设显示文字为“加密”，命名为 submitButton
5. 在窗口里生成一个标签，命名为 codeLabel
6. 定义函数 encrypt，带 event 事件参数
   1）找到文本框里的明文，存入变量 original
   2）original[::-1]将 original 里明文的字符串反转，将返回的结果存入局部变量 code
   3）设标签的文本为 code 里面的密文
7. 把 encrypt 函数绑定到点击 submitButton 按钮的事件<Button-1>上

8. 运行窗口

```
1. from tkinter import *
2. window = Tk()
3. inputBox = Entry(window)
   inputBox.pack()

4. submitButton = Button(window,text="加密")
   submitButton.pack()

5. codeLabel = Label(window)
   codeLabel.pack()

6. def encrypt(event):
   → original = inputBox.get()
   → code = original[::-1]
   → codeLabel.config(text=code)

7. submitButton.bind("<Button-1>",encrypt)
8. window.mainloop()
```

编写好后，保存、运行代码。

之后我们会了解不同的加密方法，用不同的方式编写加密函数。

11.3 案例 2：绕小弯说话——凯撒密码

11.3.1 凯撒密码介绍

凯撒密码是一种替换密码，明文里的每个字母都是用其他字母替换的。替换的规律是将字母往右移动几格。具体移动的数字我们称之为**密钥**。

相传凯撒大帝每次会将字母移动 3 格，他的密钥是 3。所以 a 会变成 d，b 会变成 e，以此类推，我们可以把这个规律总结成一个转盘，如图 11-2 所示。

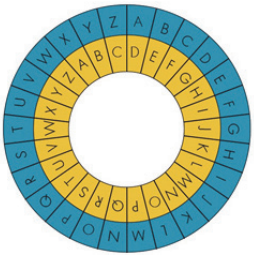


图 11–2　凯撒密码转盘

为了方便数格子，我们列出了每个字母对应的数字，也就是字母表中字符串的索引值位置。

```
alphabet = "abcdefghijklmnopqrstuvwxyz"
alphabet[0]会返回"a"
alphabet[15]会返回"p"
```

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

如果明文是“apple”，按密钥为 3 来看，对应的密文就是“dssoh”。我们找到每个字母对应的数字，加上 3，得到密文对应的数字，再找到对应数字的字母。这就是凯撒密码的加密方法。

明文	a	p	p	l	e
明文对应数字	0	15	15	11	4
密钥 3，移动 3 格	+3	+3	+3	+3	+3
密文对应数字	3	18	18	14	7
密文	d	s	s	o	h

11.3.2　编写凯撒密码

编写一段代码，让用户输入明文，再让用户输入密钥，然后根据凯撒密码的加密方法打印出密文。

实例

请输入明文>>>apple
请输入密钥>>>3
你的密文是：dssoh

凯撒密码程序 1

1. 定义函数 encrypt，带参数明文字符串 original 和密钥 shift
   1）将字母表存在变量 alphabet 里
   2）用变量 code 装密文，初始时为空字符串
   3）对于明文 original 里的每个字，用循环重复下面的代码：
      a）找到每个字在字母表里的索引值，存到 num 里
      b）给这个索引值加上密钥 shift
      c）找到这个数字除以 26 的余数，得到密文对应的数字，存到 newNum 里
      d）找到 newNum 对应的字母，添加到 code 里
   4）当 original 里的每个字母都经过 for 循环加密后，返回装密文的 code
2. 用 input()让用户输入明文，将明文字符串存入变量 text
3. 用 input()让用户输入密钥，将密匙字符串存入变量 key
4. 用 int()将 key 里的字符串转换成数字
5. 调用 encrypt 函数，将 text 变量里的明文字符串和 key 里的密钥数字作为参数，将返回的密文值存入变量 secret
6. 打印密文

```
1. def encrypt(original, shift):
   →  alphabet = "abcdefghijklmnopqrstuvwxyz"
   →  code = ""
   →  for x in original:
   →  →  num = alphabet.find(x)
   →  →  newNum = num + shift
   →  →  newNum = newNum % 26
   →  →  code = code + alphabet[newNum]
   →  return code

2. text = input("请输入明文>>>")
3. key = input("请输入密钥>>>")
4. key = int(key)
5. secret = encrypt(text, key)
6. print("你的密文是：", secret)
```

注意，这里的 newNum 有一行 newNum % 26 要处理。原因是 alphabet 里对应字母的数字只有 0 ~ 25。但我们在做加法时，偶尔会超出 25，此时我们需要重新回到 0 来数数。

例如，如果要将 w 往右移动 5 格，那么 w 移动 3 格后，第 4 格回到 a，第 5 格回到 b。按照我们的算法，22 + 5 = 27。我们用 27 % 26 可以得到 1，1 对应 b。不论数字多大，只要使用% 26，我们就可以得到 0~25 对应的数字和字母。

编写好后，保存、运行代码。看看如果输入 attack tomorrow，是否可以得到下面的结果?

```
请输入明文>>>attack tomorrow
请输入密钥>>>5
你的密文是:  fyyfhpeytrtwwtb
```

细心的人会发现，其中的空格都被 e 替代了。那是因为空格字符串“ ”不在 alphabet 字符串"abcdefghijklmnopqrstuvwxyz"里面，所以 alphabet.find(x)会返回-1，而移动 5 格后，它的 newNum 是 4，4 对应的是 e。

若要保留空格，可以在 encrypt 函数里加一个条件判断，如果是空格，那么直接将空格添加到 code，否则进行加密。在下面的代码中，如果字母表里找不到该空格，那么 alphabet.find（x）将会返回-1，表示这个字符不在 alphabet 字符串里。

```
def encrypt(original,shift):
→  alphabet = "abcdefghijklmnopqrstuvwxyz"
→  code = ""
→  for x in original:
→  →  num = alphabet.find(x)
→  →  if num == -1:
→  →  →  code = code + " "
→  →  else:
→  →  →  newNum = num + shift
→  →  →  newNum = newNum % 26
→  →  code = code + alphabet[newNum]
→  return code
```

这样，attack tomorrow 密钥是 5 的密文就会变成 fyyfhp ytrtwwtb。

```
请输入明文>>>attack tomorrow
请输入密钥>>>5
你的密文是:  fyyfhp ytrtwwtb
```

11.3.3 编写凯撒密码窗口

与之前类似，我们可以用 Tkinter 编写凯撒密码的窗口，方便大家使用，如图 11-3 所示。

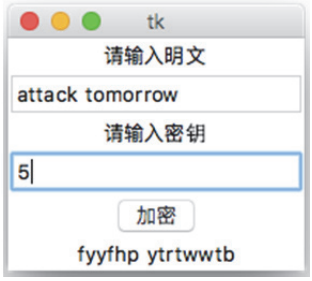


图 11-3 凯撒密码的窗口

凯撒密码程序 2

```
from tkinter import *
window = Tk()

messageLabel = Label(window,text="请输入明文")
messageLabel.pack()

inputBox = Entry(window)
inputBox.pack()

messageLabe2 = Label(window,text="请输入密钥")
messageLabe2.pack()

keyBox = Entry(window)
keyBox.pack()

submitButton = Button(window,text="加密")
submitButton.pack()

codeLabe1 = Label(window)
codeLabe1.pack()

def encrypt(event):
    alphabet = "abcdefghijklmnopqrstuvwxyz"
    original = inputBox.get()
    key = int(keyBox.get())
    code = ""
```

```
    for x in original:
        num = alphabet.find(x)
        if num == -1:
            code = code + " "
        else:
            newNum = num + key
            newNum = newNum % 26
            code = code + alphabet[newNum]
    codeLabel.config(text=code)

submitButton.bind("<Button-1>", encrypt)

window.mainloop()
```

编写好后，保存、运行代码，看看是否可以得到预想中的结果。

11.3.4　破解凯撒密码

凯撒密码其实也比较容易破解，只要把 1~25 每个密钥试一遍，就可以找到 25 种组合，推测出正确的组合。例如，对于 DSSOH 这行密文，最多只要移动 25 次，就可以找出合理的破译方法。

ETTPI	KZZVO	QFFBU	WLLHA	CRRNG
FUUQJ	LAAWP	RGGCV	XMMIB	
GVVRK	MBBXQ	SHHDW	YNNJC	
HWWSL	NCCYR	TIIEX	ZOOKD	
IXXTM	ODDZS	UJJFY	APPLE ✔	
JYYUN	PEEAT	VKKGZ	BQQMF	

11.4　案例 3：混乱着说话——打乱替换密码

11.4.1　打乱替换密码介绍

凯撒密码可以很轻易地被破解。要想增加破解难度，我们可以打乱每个替换的字母，排成一个由 26 个字母组成的密钥。

例如，我们的密钥是以下一连串打乱的字母。

"kpzxleoryafiqjcbsvnwtdhmug"

alphabet　= "abcdefghijklmnopqrstuvwxyz"

key = "kpzxleoryafiqjcbsvnwtdhmug"

a 对应 k，b 对应 p，以此类推。

alphabet[0]加密成 key[0]，alphabet[1]加密成 key[1]等。

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
k	p	z	x	l	e	o	r	y	a	f	i	q
n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25
j	c	b	s	v	n	w	t	d	h	m	u	g

根据这个密钥，若明文是"apple"，a 对 k，p 对 b，l 对 i，e 对 l，则“apple”会被加密成“kbbil”。

明文	a	p	p	l	e
对应数字	0	15	15	11	4
密文	k	b	b	i	l

11.4.2 编写打乱替换密码

编写一段代码，让用户输入明文，再让用户输入密钥的字符串，然后打印出密文。

```
请输入明文>>>apple
请输入 26 个字母的密钥>>>kpzxleoryafiqjcbsvnwtdhmug
你的密文是：kbbil
```

打乱替换密码程序

1. 定义函数 encrypt，带参数明文字符串 original 和密钥 key
   1）将字母表存在变量 alphabet 里
   2）用变量 code 装密文，初始时为空字符串
   3）对于明文 original 里的每个字，用 for 循环重复下面的代码：
      a）先找到每个字在字母表里的索引值，存进 num 里
      b）如果 num 是-1，表示它不在 26 个字母里，则直接把原文添加进密文 code 里
      c）否则用 num 为索引值找到密钥 key 里对应的字母，添加进密文 code 里
   4）当 original 里的每个字母都经过 for 循环加密后，返回装密文的 code
2. 用 input()让用户输入明文，将明文字符串存进变量 text 里
3. 用 input()让用户输入密钥打乱的 26 个字母，将密钥字符串存进变量 myKey 里
4. 调用 encrypt 函数，将 text 变量里的明文字符串和 myKey 里的密钥数字作为参数，将返回的密文值存

进变量 secret 里

5. 打印密文

```
def encrypt(original, key):
    alphabet = "abcdefghijklmnopqrstuvwxyz"
    code = ""
    for x in original:
        num = alphabet.find(x)
        if num == -1:
            code = code + x
        else:
            code = code + key[num]
    return code

text = input("请输入明文>>>")
myKey = input("请输入 26 个字母的密钥>>>")
secret = encrypt(text,myKey)
print("你的密文是：",secret)
```

编写好后，保存、运行代码。看看如果输入 attack tomorrow，是否可以得到下面的结果?

编写好简单的程序后，你是否可以尝试用 Tkinter 为这个加密程序添加一个如图 11-4 所示的前端窗口?

自己先练习一下吧，参考答案见本书附赠资料。

```
请输入明文>>>attack tomorrow
请输入 26 个字母的密钥>>> kpzxleoryafiqjcbsvnwtdhmug
你的密文是：kwwkzf wcqcvvch
```

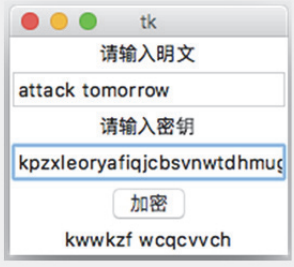


图 11-4　前端窗口

11.5　案例 4：绕大弯说话——维吉尼亚密码

11.5.1　维吉尼亚密码介绍

凯撒密码的密钥是 0~25 的一个数字，所以只要试 25 次，就可以试出明文，快速破译。维吉

尼亚密码在凯撒密码的基础上，提出用一组数字 0~25 作为密钥。具体的密钥是一个单词。在用凯撒密码加密时，我们每次移动的格子数是固定的。而在用维吉尼亚密码加密时，我们会根据密钥对应的数字，每次移动不同的格子数。

如果我的密钥是“sky”，它对应的一组数字就是(18,10,24)。

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

加密时的过程如下：

第 1 个字母往右移 18 格

第 2 个字母往右移 10 格

第 3 个字母往右移 24 格

第 4 个字母往右移 18 格

第 5 个字母往右移 10 格

第 6 个字母往右移 24 格

……

如果明文是“apple”，以密钥为“sky”来看，对应的密文则是“szndo”。找到每个字母对应的数字，依次加上 18、10、24，得到密文对应的数字，再找到对应数字的字母。

明文	a	p	p	l	e
明文对应数字	0	15	15	11	4
密钥“sky” 移动 18、10、24 格	+18	+10	+24	+18	+10
密文对应数字	18	25	39 → 13	29 → 3	14
密文	s	z	n	d	o

11.5.2 编写维吉尼亚密码

手动给明文加密需要进行大量计算，耗时长且容易出错，这时我们就需要用程序来加密和解密。

编写一段代码，让用户输入明文及密钥单词，然后打印密文。

```
请输入明文>>>apple
请输入密钥单词>>>sky
你的密文是：szndo
```

维吉尼亚密码程序

1. 定义 encrypt 函数，带参数明文的字符串 original 和密钥 key
   1）将字母表存在变量 alphabet 里
   2）用 listOfNumbers 装密钥对应的一串数字
   3）用变量 code 装密文，初始时为空字符串
   4）对于密钥里的每个字，用 for 循环重复下面的代码：
      a）找寻每个字母对应的数字，存入 keyNum
      b）将这个数字添加到 listOfNumbers 里
   5）keyIndex 代表我们使用 listOfNumbers 里面的第几个数字为移动的格子数。从 0 开始，每次移动完后，keyIndex 加 1，如果到了列表末尾，keyIndex 回到 0
   6）对于明文 original 里的每个字，用 for 循环重复下面的代码：
      a）找到这个字在字母表里的索引值，存进 num 里
      b）如果 num 是-1，表示它不在 26 个字母里，则直接把原文添加进密文 code 里
      c）否则用 keyIndex 找到 listOfNumbers 里面对应的数字，加上 num，成为移动后的新索引值 newNum。找到 newNum 除以 26 的余数，存进 newNum 里，用 newNum 为索引值找到对应的字母，添加进密文 code 里
   7）每次移动完后，keyIndex 加 1，如果到了列表末尾，keyIndex 就回到 0
   8）完成后返回密文
2. 用 input()让用户输入明文，将明文字符串存进变量 text 里
3. 用 input()让用户输入密钥单词，存进变量 myKey 里
4. 调用 encrypt 函数，将 text 变量里的明文字符串和 myKey 里的密钥作为参数，将返回的密文值存进变量 secret 里
5. 打印密文

```
1. def encrypt(original,key):
   →  alphabet = "abcdefghijklmnopqrstuvwxyz"
   →  listOfNumbers = []
   →  code = ""

   →  for letter in key:
   →  →  keyNum = alphabet.find(letter)
   →  →  listOfNumbers.append(keyNum)

   →  keyIndex = 0
   →  for x in original:
   →  →  num = alphabet.find(x)
   →  →  if num == -1:
```

```
            code = code + " "
        else:
            newNum = num + listOfNumbers[keyIndex]
            newNum = newNum % 26
            code = code + alphabet[newNum]
        if keyIndex < len(listOfNumbers) - 1:
            keyIndex += 1
        else:
            keyIndex = 0

    return code

text = input("请输入明文>>>")
myKey = input("请输入密钥单词>>>")
secret = encrypt(text,myKey)
print("你的密文是：",secret)
```

编写好后，保存、运行代码。看看如果输入 attack tomorrow，使用密钥单词 kangaroo，是否可以得到下面的结果？

```
请输入明文>>> attack tomorrow
请输入密钥单词>>> kangaroo
你的密文是：ktggcb hymbxrfk
```

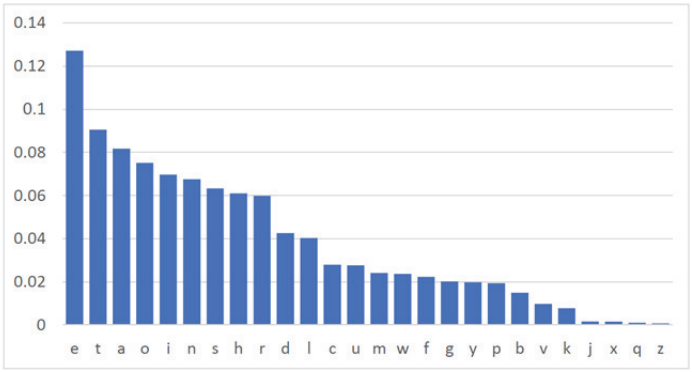
11.6 案例 5：靠计数破译密码

如果你收到下面这样一段密文，你要如何破解呢？

yw hkn wrl plnw ce wyqln, yw hkn wrl hcvnw ce wyqln, yw hkn wrl kol ce hynxcq, yw hkn wrl kol ce ecciynrjlnn, yw hkn wrl lbczr ce pliyle, yw hkn wrl lbczr ce yjzvlxtiywu, yw hkn wrl nlkncj ce iyorw, yw hkn wrl nlkncj ce xkvfjlnn, yw hkn wrl nbvyjo ce rcbl, yw hkn wrl hyjwlv ce xlnbkyv.

事实上在大部分语言里，字母出现的次数非常不同。比如在英文里，E、T、A、O、I 出现得最频繁，而 Z、Q、X、J、K 出现得最不频繁。

英文字母出现频率：



我们可以编写一段程序，帮我们数每个字母出现的次数。根据密文里字母出现的频率，找到对应的明文字母。

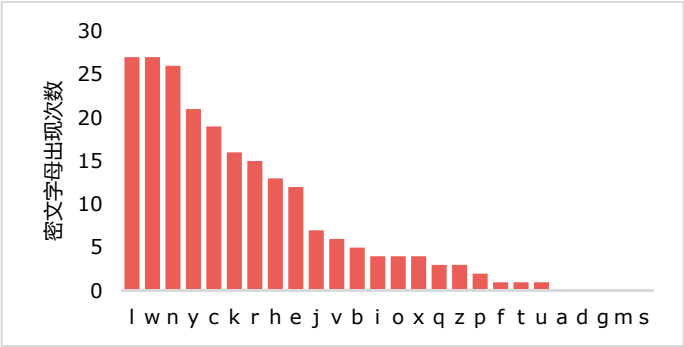
计算字母出现频率的代码

1. 将字母表存在变量 alphabet 里
2. 将要破译的密码存在变量 code 里
3. 新建一个字典叫 letterCount，用来计算每个字母出现的次数，开始时将所有字母都设为 0
4. 用 for 循环针对 code 里的每个字母，如果它是 alphabet 里的 26 个字母之一，则让它在 letterCount 字典里的值增加 1
5. 循环完后打印统计资料

```
1. alphabet = "abcdefghijklmnopqrstuvwxyz"
2. code = "yw hkn wrl plnw ce wyqln, yw hkn wrl hcvnw ce wyqln, yw hkn wrl kol ce
   hynxcq, yw hkn wrl kol ce ecciynrjlnn, yw hkn wrl lbczr ce pliyle, yw hkn wrl lbczr
   ce yjzvlxtiywu, yw hkn wrl nlkncj ce iyorw, yw hkn wrl nlkncj ce xkvfjlnn, yw hkn
   wrl nbvyjo ce rcbl, yw hkn wrl hyjwlv ce xlnbkyv."
3. letterCount = {'a': 0, 'b': 0, 'c': 0, 'd': 0, 'e': 0, 'f': 0, 'g': 0, 'h': 0, 'i': 0,
   'j': 0, 'k': 0, 'l': 0, 'm': 0, 'n': 0, 'o': 0, 'p': 0, 'q': 0, 'r': 0, 's': 0, 't': 0,
   'u': 0, 'v': 0, 'w': 0, 'x': 0, 'y': 0, 'z': 0}
4. for letter in code:
   →  if letter in alphabet:
   →  →  letterCount[letter] += 1

5. print(letterCount)
```

制成表格后，统计资料如下：



根据计数结果，密文里 l 和 w 最常见，它们对应着明文里的 e 和 t，尝试用绿色的 e 和 t 替换 l 和 w，得到下面的结果。

yt hkn tre pent ce tyqen, yt hkn tre hcvnt ce tyqen, yt hkn tre koe ce hynxcq, yt hkn tre koe ce ecciynrjenn, yt hkn tre ebczr ce peiyee, yt hkn tre ebczr ce yjzvextiytu, yt hkn tre neknej ce iyort, yt hkn tre neknej ce xkvfjenn, yt hkn tre nbvyjo ce rcbe, yt hkn tre hyjtev ce xenbkyv.

it 和 the 在英文中也非常常见。根据观察，上文里的 yt 有大概率是 it，而 tre 有大概率是 the。对应 y 的应该就是 i，对应 r 的应该就是 h。替换这两个字母，得到下面的结果。

it hkn the pent ce tiqen, it hkn the hcvnt ce tiqen, it hkn the koe ce hinxcq, it hkn the koe ce ecciinhjenn, it hkn the ebczh ce peiiee, it hkn the ebczh ce ijzvextiitu, it hkn the neknej ce iioht, it hkn the neknej ce xkvfjenn, it hkn the nbvijo ce hcbe, it hkn the hijtev ce xenbkiv.

hkn 出现多次，前面是 it，后面是 the，hkn 有大概率是 was。ce 这个组合也出现多次，有大概率是 of。我们将 h、k、n、c、e 分别用 w、a、s、o、f 替换，看看能得到什么结果。

it was the pest of tiqes, it was the wovst of tiqes, it was the aoe of wisxoq, it was the aoe of fooiishjess, it was the ebozh of peiief, it was the ebozh of ijzvextiitu, it was the seasoj of iioht, it was the seasoj of xavfjess, it was the sbvijo of hobe, it was the wijtev of xesbaiv.

这样已经比较清楚明了了，我们可以再进一步分析。

fooiishjess 有大概率是 foolishness，seasoj 有大概率是 season。这样基本可以确定 j 是 n 且 i 是 l。将它们代入原文。

wisxom 有大概率是 wisdom，那么 x 是 d，代入原文。

pest 可能是 best 或者 pest。aoe 可能是 ape 或者 age，应该不是 ate。tiqes 可能是 times 或者 tides。我们先假设 p 是 b，o 是 g，q 是 m，代入原文看看：

it was the best of times, it was the wovst of times, it was the age of wisdom, it was the age of foolishness, it was the ebozh of belief, it was the ebozh of inzvedtlitu, it was the season of light, it was the season of davfness, it was the sbving of hobe, it was the wintev of desbaiv.

e	t	a	o	i	n	s	h	r	d	l	c	u
l	w	k	c	y	j	n	r	?	x	i		
m	w	f	g	y	p	b	v	k	j	x	q	z
q	h	e	o			p						

在密文频率排名的前十位 lwnyckrhejvb 中，lwnyckrhej 分别对应 etsioahwfn。如果下一次轮到 v 时还没有对应的字母，则可以尝试对应 r。这样 wintev 会变成 winter，wovst 会变成 worst，这样比较合理。

it was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the ebozh of belief, it was the ebozh of inzredtlitu, it was the season of light, it was the season of darfness, it was the sbring of hobe, it was the winter of desbair.

到这一步，我们的密码基本就破解出来了：

it was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of light, it was the season of darkness, it was the spring of hope, it was the winter of despair.

这是英国作家狄更斯在小说《双城记》里的开头语：

这是最好的时代，这是最坏的时代，这是智慧的时代，这是愚蠢的时代，这是信仰的时期，这是怀疑的时期，这是光明的季节，这是黑暗的季节，这是希望之春，这是失望之冬。

用一些逻辑和编程的知识，再加上对语言的了解，认真观察规律，破解简单的密码并非难事。

除破解密码外，你还可以用频率分析来研究古诗词和文学作品。

11.7　总结及课后练习

在这一章里，我们了解了不同的加密方法，并用代码编写了相关的程序。最后，我们还了解了如何通过频率分析破译简单的替换密码。

1. 在加密、解密、破译密码的过程中，相比用手计算，用计算机计算有哪些优点呢？请讨论。

2. 我们了解了多种英文加密方法，你知道中文中有哪些不同的加密方法呢？请上网查找了解。

3. 请用凯撒密码，密钥为 12，加密下面的信息：

   the navy is coming

4. 请破解下面的凯撒密码：

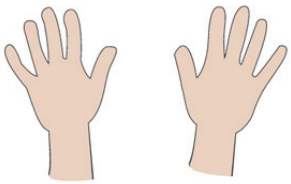
   uan plpn cdl

第 12 章
二进制数的世界

本章将介绍二进制数，以及为什么计算机世界都用二进制数。还会介绍图片编码及字符编码 ASCII 编码，并编写二进制数和十进制数转换的函数。

12.1　二进制数是什么

计算机世界有 10 种“人”，即会二进制数的人和不会二进制数的人。



我们习惯于十进制数的世界，用 0、1、2、3、4、5、6、7、8、9 这 10 个数字来代表我们的数字，正好对应我们的十根手指头。

但是计算机的世界是二进制数的世界，用 0 和 1 这两个数字代表所有的信息。我们发送的信息、手机程序、照片、视频都是用二进制数来存储的。

为什么计算机不用我们熟悉的十进制数呢?

如果要用十进制数，我们计算机里的硬件得识别 10 种不同的状态，对应 10 个不同的数字，难以做到。而识别 2 种状态就简单得多，比方说一个电路开关，打开代表 0，合上代表 1，如图 12-1 所示。多个开关组合起来，可以表达各种各样的数据信息。

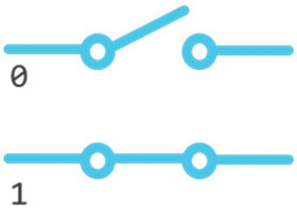


图 12-1　二进制数符合电子开关状态

二进制数	十进制数	二进制数	十进制数
0001	1	0101	5
0010	2	0110	6
0011	3	0111	7
0100	4	1000	8

回到这一小节开头的那个脑筋急转弯：这个世界有 10 种人 => 这个世界有两种人，会二进制数的人和不会二进制数的人。

10 在二进制数里代表数字“二”。

12.2 二进制数转十进制数

了解二进制数以后，我们来思考如何在二进制数和十进制数之间互相转换。首先思考一下我们熟悉的十进制数，它有个位、十位、百位、千位。

例如 234 的个位是 4，十位是 3，百位是 2，代表它是 4 个 1，3 个 10 和 2 个 100 加起来得到的数字。

2	3	4
百位	十位	个位
10 的 2 次方	10 的 1 次方	10 的 0 次方
100	10	1

234 = 2 x 100 + 3 x 10 + 4 x 1

类似十进制数，二进制数也有不同的数位，不过是个位、二位、四位、八位、十六位等。

例如 11010011 的个位是 1，二位是 1，四位是 0，八位是 0，十六位是 1，三十二位是 0，六十四位是 1，一百二十八位是 1，代表它是 1 个 1，1 个 2，1 个 16，1 个 64 和 1 个 128 加起来得到的数字。

1	1	0	1	0	0	1	1
一百二十八位	六十四位	三十二位	十六位	八位	四位	二位	个位
2 的 7 次方	2 的 6 次方	2 的 5 次方	2 的 4 次方	2 的 3 次方	2 的 2 次方	2 的 1 次方	2 的 0 次方
128	64	32	16	8	4	2	1

1 x 128 + 1 x 64 + 0 x 32 + 1 x 16 + 0 x 8 + 0 x 4 + 1 x 2 + 1 x 1

= 211

编程练习：请将下面的二进制数转成十进制数。

1. 11010010

2. 01010101

3. 10011001

编写一段代码，让用户输入的二进制数字符串转换成十进制数字符串。

实例

```
请输入二进制数字>>>11010011
211
```

二进制数转十进制数 1

1. 用 input()让用户输入二进制数，存进变量 num2
2. 用 int(num2, 2)将二进制数字符串转换成十进制数字符串，存进变量 num10
3. 打印 num10 里的数字

```
1. num2 = input("请输入二进制数>>>")
2. num10 = int(num2,2)
3. print(num10)
```

Python 自带二进制数和十进制数转换的函数：

- 用 int("101",2)可以把字符串"101"代表的二进制数转换成十进制数
- 用 bin(5)可以将十进制数 5 转换成二进制数 101

如果想要挑战自己，也可以尝试自己写出转换的程序，代码如下：

二进制数转十进制数 2

1. 用 input()让用户输入二进制数，存进变量 num2
2. 将字符串调转，存进 num2Reverse 里，这样我们就可以先处理个位数
3. 用变量 num10 保存我们的十进制数答案，先让它为 0
4. 用 index 表示我们在处理第几位数。我们从 0 开始，也就是 2 的 0 次方，个位数
5. 针对 num2Reverse 里的每一位数，重复下面的代码：
   1）如果它是 1，则找到它是几位数，将这位数加进 num10 里
   2）index + 1，进入下一个数位
6. 循环后打印 num10 里存的十进制数

```
1. num2 = input("请输入二进制数字>>>")
2. num2Reverse = num2[::-1]
3. num10 = 0
4. index = 0
```

```
5. for i in num2Reverse:
       if i == "1":
           num10 = num10 + 2 ** index
       index += 1

6. print(num10)
```

我们来讲讲这个程序的逻辑，以转换 10011 为例：

num2 = "10011"

num2Reverse = "11001"

调转位置我们可以先处理个位数。

num2Reverse 字符串里有 5 个字符，所以循环 5 次。每次循环时，如果值是 1，则找到它的数位，将数位添加进十进制数，存在 num10 里。例如二进制数 10011 的个位、二位、十六位是 1，那它的十进制数是 1 + 2 + 16 = 19。

下表展示了几个变量的循环过程。

循环次数	循环前 num10	循环前 index	i	2 ** index	i == “1”	循环后 num10	循环后 index	计算
1	0	0	"1"	1	True	1	1	1 x 1
2	1	1	"1"	2	True	3	2	1 x 2
3	3	2	"0"	4	False	3	3	0 x 4
4	3	3	"0"	8	False	3	4	0 x 8
5	3	4	"1"	16	True	19	5	1 x 16

二进制数 10011 对应的十进制数是 19。

12.3 十进制数转二进制数

若要将十进制数转成二进制数，则可以将一个数字除以 2，用每次得到的余数组成二进制数。

例如将十进制数 211 转成二进制数，我们将 211 除以 2，得 105 余 1，余数是个位上的数字。再用 105 除以 2，得 52 余 1，余数是第二位数字，也就是二位上的数字。再用 52 除以 2，得 26 余 0，余数 0 是第三位数字，也就是四位上的数字。以此类推，直到 0 为止。

211	除以 2	105	余	1	个位
105	除以 2	52	余	1	二位
52	除以 2	26	余	0	四位
26	除以 2	13	余	0	八位
13	除以 2	6	余	1	十六位
6	除以 2	3	余	0	三十二位
3	除以 2	1	余	1	六十四位
1	除以 2	0	余	1	一百二十八位
0 停止					

将余数从下往上读，组成二进制数 11010011。

编程挑战：尝试编写十进制数转二进制数的代码。

12.4　图片都是数字

在信息世界里，所有的数据最终都是用 1 和 0 来表示的。

屏幕上的图片由小方格组成，这些小方格叫像素。同样一张图片，格子越小、越多，像素就越高，看起来也更清晰，如图 12 -2 所示。

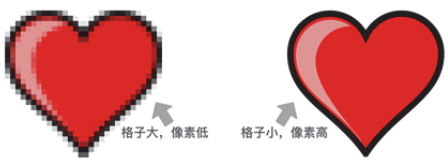


图 12-2　高像素和低像素的区别

每个像素有对应的颜色，而这种颜色是用数字来表示的。

- 1 个二进制数的数位可以描述 2 种颜色，0 是黑，1 是白
- 2 个二进制数的数位可以描述 4 种颜色，00、01、10、11
- 3 个二进制数的数位可以描述 8 种颜色，000、001、010、011、100、101、110、111

过去常用的 8 位图用 8 个二进制数的数位描述 2 的 8 次方，也就是 256 种颜色。而现在常用的 RGB 颜色模型位图，用 24 个二进制数的数位描述 2 的 24 次方，也就是 16777216 种颜色。

图 12 -3 中的粉红色格子用 111001101001010110010101 表示，深红色格子用 11101000 00111110110011 表示。每个格子都用一串 24 位的二进制数表示，这些 1 和 0

组成了如图 12 -3 所示的图片。

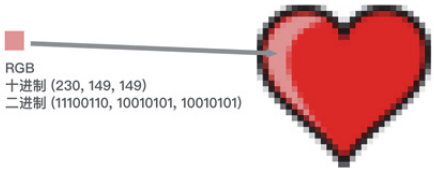


图 12-3　用数字表示颜色

12.5　字母都是数字

上一章我们介绍了如何给文字加密。事实上，在计算机的世界里，为了方便文字的存储和传输，文字也会用 1 和 0 来表示。

而具体哪个字符用什么数字表示，就涉及具体的编码规则了。为了方便沟通，大家约定了一套通用的编码规则，称为 ASCII 编码。

二进制编码	十进制编码	代表字符
00100000	32	(空格)
00100001	33	!
00100010	34	"
00100011	35	#
00100100	36	$
00100101	37	%
00100110	38	&
00100111	39	'
00101000	40	(
00101001	41	)
00101010	42	*
00101011	43	+
00101100	44	,
01011011	91	[
01011100	92	\

二进制编码	十进制编码	代表字符
01000001	65	A
01000010	66	B
01000011	67	C
01000100	68	D
01000101	69	E
01000110	70	F
01000111	71	G
01001000	72	H
01001001	73	I
01001010	74	J
01001011	75	K
01001100	76	L
01001101	77	M
01001110	78	N
01001111	79	O

二进制编码	十进制编码	代表字符
01100001	97	a
01100010	98	b
01100011	99	c
01100100	100	d
01100101	101	e
01100110	102	f
01100111	103	g
01101000	104	h
01101001	105	i
01101010	106	j
01101011	107	k
01101100	108	l
01101101	109	m
01101110	110	n
01101111	111	o

续表

二进制编码	十进制编码	代表字符
01011101	93	]
01011110	94	^
01011111	95	_
01100000	96	`

二进制编码	十进制编码	代表字符
01010000	80	P
01010001	81	Q
01010010	82	R
01010011	83	S
01010100	84	T
01010101	85	U
01010110	86	V
01010111	87	W
01011000	88	X
01011001	89	Y
01011010	90	Z

二进制编码	十进制编码	代表字符
01110000	112	p
01110001	113	q
01110010	114	r
01110011	115	s
01110100	116	t
01110101	117	u
01110110	118	v
01110111	119	w
01111000	120	x
01111001	121	y
01111010	122	z

这里有一段编码后的文字，如下：

```
01000011011011110110110101100101001000000110001001100001011000110110101100100000
01101000011011110110110101100101
```

根据 ASCII 解码，分析如下：

01000011 代表 C

01101111 代表 o

01101101 代表 m

01100101 代表 e

00100000 代表（空格）

01100010 代表 b

01100001 代表 a

01100011 代表 c

01101011 代表 k

00100000 代表（空格）

01101000 代表 h

01101111 代表 o

01101101 代表 m

01100101 代表 e

所以这串二进制数组成了 Come back home 这句话。

细心的读者会发现大写 C 是 01000011，小写 c 是 01100011，它们的结尾相似，这大概是设计者用心巧妙之处吧。

12.6 总结及课后练习

在这一章里，我们简单了解了为什么数字世界习惯用二进制数，以及数字世界是如何用数字表示不同内容的。我们还学习了二进制数和十进制数之间的转换。

1. 根据 ASCII 编码规则，请将 STOP!转换成一串二进制数。

2. 请将下面的十进制数转换成二进制数。
   a）35
   b）89
   c）234

3. 请将下面的二进制数转换成十进制数。
   a）10010
   b）11111
   c）110110

4. 我们在这一章里了解了二进制数。事实上，我们常用的还有十六进制数，请上网查找相关资料，了解十六进制数是由哪 16 个字符组成的？它有哪些用处？

第 13 章

潜水钟与蝴蝶——用计算思维解决问题

本章将以故事的形式介绍编程和计算思维在现实生活中的意义，启发读者进一步探索编程和计算机科学。

13.1　潜水钟与蝴蝶的故事

如果你的朋友被“冰封”在自己的身体里，你要如何帮助他?

这句话听上去像是魔幻小说中的故事，但世界上的确有一种可怕的疾病，叫闭锁综合征。如果你的朋友不幸罹患闭锁综合征，那么他很可能全身瘫痪，只剩下眼睛可以眨。

他就像被冰封在自己的身体里一样，可以听得到、看得到、有思想、有情感，却动弹不得。

如果你的朋友患了闭锁综合征，你要如何帮助他？我们可以学习医学和科学，等待科学的进步能够治愈闭锁综合征。不过每一分钟的等待对他们来讲都是痛苦的，而科学研究的时间却非常漫长。

我们学习了编程和计算机科学后，如何用我们所学的知识帮助他们与外界沟通呢?

事实上，英国物理学家霍金也患有闭锁综合征，他全身只有脸颊的一块肌肉可以动。他用一块肌肉完成了许多科学研究，写了多本物理书。法国作家让 · 多米尼克 · 鲍比也患有闭锁综合征，只有左眼皮可以动，他用左眼皮完成了小说《潜水钟与蝴蝶》，介绍“冰封”在自己身体里的感受。

他们是怎么做到的?

13.1.1 第一次尝试——眨眼次数代表的字母

我们来思考一下，如何帮助被“冰封”的朋友，让他能够与外界交流，从跟护士和家人沟通，到进一步写小说、做学问，让他在“冰封”的身体里也活得有意义。

我们可以设计一个系统，告诉他眨 1 下眼，代表 a，眨 2 下眼，代表 b，以此类推。只要记录他眨了几下眼，就可以知道他说了什么话。

1	2	3	4	5	6	7	8	9	10	11	12	13
a	b	c	d	e	f	g	h	i	j	k	l	m
14	15	16	17	18	19	20	21	22	23	24	25	26
n	o	p	q	r	s	t	u	v	w	x	y	z

例如下面这个 Python 的 blinks(n)函数，只要输入眨眼次数 n，就可以输出他要说的话。

```
def blinks(n):
→  alphabet = "_abcdefghijklmnopqrstuvwxyz"
→  print("他说" + alphabet[n])
```

```
>>> blinks(8)
他说 h
>>> blinks(5)
他说 e
>>> blinks(12)
他说 l
>>> blinks(12)
他说 l
>>> blinks(15)
他说 o
```

可是，仅“hello”就要眨 8 + 5 + 12 + 12 + 15 = 52 次眼，我们的朋友会很累，更何况我们还没有加标点符号。因此，我们可以想一下如何改进这个系统。

13.1.2 第二次尝试——二分搜索

我们可以改进这个系统，眨眼睛是 True，不眨眼睛是 False。然后连续提问，搜索他想说的话。例如下面这个例子：

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

你想要的字母在 a～m 里面吗？

眨眼睛 True:

a b c d e f g h i j k l m n o p q r s t u v w x y z

你想要的字母在 a～g 里面吗?

不眨眼睛 False:

a b c d e f g h i j k l m n o p q r s t u v w x y z

你想要的字母在 h～j 里面吗?

眨眼睛 True:

a b c d e f g h i j k l m n o p q r s t u v w x y z

你想要的字母在 h～i 里面吗?

眨眼睛 True:

a b c d e f g h i j k l m n o p q r s t u v w x y z

你想要的字母在 h～h 里面吗?

眨眼睛 True:

a b c d e f g h i j k l m n o p q r s t u v w x y z

只要经历 5 轮眨眼或不眨眼，就可以找到字母 h。这里用到了计算机科学中有名的二分查找法，每次排除 1/2 的可能性，如图 13-1 所示。

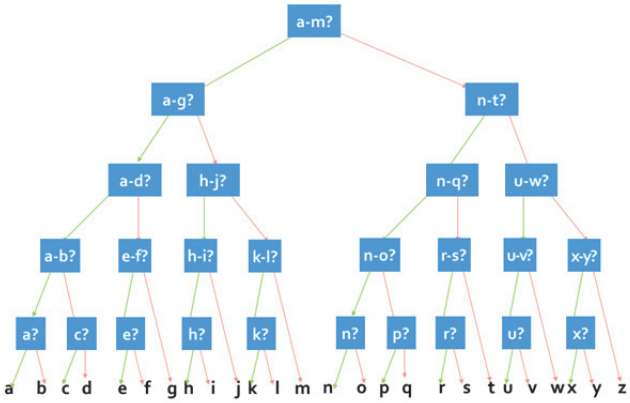


图 13-1　通过二分查找法快速找到患者想要表达的字母

我们可以把代码写出来，试验一下，假设眨眼是 True，不眨眼是 False，输入用户 5 次眼部

的动作，编写程序解读患者想要表达的字母。

```
def translate(blinks):
→  alphabet = "abcdefghijklmnopqrstuvwxyz"
→  min = 0
→  max = 25
→  for i in range(5):
→  →  min,max = check(min,max,blinks[i])
→  print(alphabet[min])

def check(min,max,blink):
→  if blink:
→  →  max = (min + max) // 2
→  else:
→  →  min = (min + max) // 2 + 1
→  return min, max
```

```
>>> translate([True,False,True,True,True])
h
>>> translate([True,True,False,True,True])
e
>>> translate([True,False,False,True,False])
l
>>> translate([True,False,False,True,False])
l
>>> translate([False,True,True,True,False])
o
```

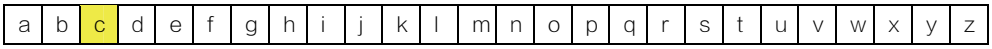
以这种方法，每次最多眨 5 次眼就好了。若要说“hello”，只需要眨眼 25 次。已经进步很多了，可是这样眨眼还是很累。我们是否还有进一步改进的方法?

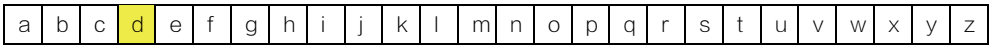
13.1.3 持续地尝试

或许我们可以让鼠标光标不断地从 a 闪烁到 z，每次定位到他想说的字母时眨眼就好了，这样每个字母只需要眨 1 次眼。

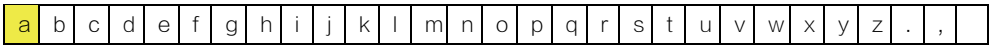
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

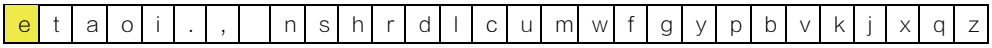




也可以把空格和标点符号加进来。



还可以根据一个字母是否常用来重新安排字母表。



或许我们可以设计系统，让他在输入“hell”的时候，自动提示“hello”。恭喜你，你已经像设计手机输入法的工程师那样开始思考了！

13.2　编写程序为身边的人解决问题

在本书的最后留给大家这样一个故事，是希望读者能够思考编程学习，学习科学技术究竟是为了什么。

希望大家学习编程不仅是学习语法，更重要的是通过学习编程认识我们这个以计算机为核心的信息世界，利用自己学会的知识来帮助身边的人解决问题。

从现在起，你可以以一名程序员的眼光看世界，发现身边需要解决的问题，静下心来研究这个问题，为它设计一个解决方案。当然，如果你的解决方案里包含 Python 编程，那就再好不过了！

本书特色

- 72个 实例
- 75个 编程练习
- 近30个 编程相关小贴士
- 19个 配套教学视频
- 1份 Python 3速查表

作者介绍

周安琪，毕业于英国剑桥大学工程系，“梦想编程”创始人。在美国、新加坡、澳大利亚和中国有多年的少儿编程教学经验，注重于通过设计有趣的编程学习体验，启发学生探索科技世界的兴趣及锻炼逻辑思维能力。

本书配套教学视频和习题参考答案下载地址
http://www.broadview.com.cn/35461

责任编辑：李利健 （QQ：41633914）
封面设计：侯士卿

上架建议：计算机 > 软件开发 > Python

ISBN 978-7-121-35461-8

定价：79.90元